VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KLIENT-SERVER MOBILNÍ APLIKACE SE ZPRACOVÁNÍM OBRAZU

CLIENT-SERVER MOBILE APPLICATION WITH IMAGE PROCESSING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. BEDŘICH ČERNOŠEK**
AUTHOR

VEDOUCÍ PRÁCE **prof. Dr. Ing. PAVEL ZEMČÍK**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/17951/2017/xcerno19

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Černošek Bedřich, Bc.**
Obor: Informační systémy
Téma: **Klient-server mobilní aplikace se zpracováním obrazu**
Client-Server Mobile Application with Image Processing
Kategorie: Zpracování obrazu

Pokyny:

1. Seznamte se s metodami tvorby klient-server aplikací s mobilním klientem (mobilní telefon, embedded systém apod.), zaměřte se na (kryptografické) ověřování času a zdroje dat, zejména obrazu.
2. Vytvořte koncepci vhodné klient-server aplikace s mobilními klienty tak, aby pracovala s obrazovými daty a umožnila ověření zdroje a času pořízení takových dat.
3. Diskutujte vlastnosti vytvořené koncepce a rozeberte výhody a nevýhody.
4. Implementujte aplikaci a demonstrujte její funkčnost na vhodném příkladě.
5. Diskutujte dosažené výsledky a možnosti pokračování práce.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1-3 zadání

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zemčík Pavel, prof. Dr. Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem této práce je vytvoření klient-server mobilní aplikace, která zahrnuje zpracování obrazu a kryptografické ověření zdroje a času vytvoření obrazu. Práce se zaměřuje na vytvoření mobilní klientské aplikace na platformě Android, která pomocí fotoaparátu mobilního zařízení bezpečně pořizuje fotografie, provádí jejich zpracování a získává digitální podpis, časové razítko a GPS souřadnice. Důležitou částí práce je bezpečná výměna klíčů, šifrovaná komunikace, datová a energetická nenáročnost klient-server aplikace. Serverová aplikace je implementována na platformě Java EE a provádí zpracování přijatého obrazu, detekci objektů, rozpoznání objektů v obraze a zajišťuje časové razítko od důvěryhodného serveru. Fotografie může být pak považována za důvěryhodný dokument použitelný jako právně platný důkazní materiál pro soudní či správní řízení.

Abstract

The main goal of this work is creating client-server application with image processing and cryptographic verification of image source and creation time. The work focuses on creating a mobile client application on the Android platform that securly takes photos by mobile device camera, processes captured images and provides a digital signature, timestamp and GPS location. The main part of the work is secure key exchange, encrypted communication, data and energy efficiency of the client-server application. The server application is implemented on the Java EE platform and processes the received image, performs object detection, object recognition in the image and provides a timestamp from a trusted server. Then taken photo can be considered as a trusted electronic document usable as valid evidence for the judical or administrative proceedings.

Klíčová slova

android aplikace, architektura klient-server, zpracování obrazu, detekce objektu v obraze, rozpoznávání objektu v obraze, kryptografie, elektronický podpis, elektronické časové razítko, šifrování, bezpečnost přenosu dat, důvěryhodný elektronický dokument

Keywords

android application, client-server architecture, image processing, object detection, object recognition, cryptography, digital signature, digital timestamp, encryption, data transfer security, trusted electronic document

Citace

ČERNOŠEK, Bedřich. *Klient-server mobilní aplikace se zpracováním obrazu.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Dr. Ing. Pavel Zemčík

Klient-server mobilní aplikace se zpracováním obrazu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana prof. Dr. Ing. Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Bedřich Černošek
21. května 2018

Poděkování

Chtěl bych poděkovat vedoucímu této práce panu prof. Dr. Ing. Pavlu Zemčíkovi za odbornou pomoc. Také bych chtěl poděkovat rodině za podporu při studiu a tvorbě této práce.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá řešením problematiky klient-server aplikace s mobilními klienty, která bezpečně vytváří fotografie pomocí fotoaparátu mobilního zařízení, provádí zpracování získaného obrazu a ověřuje zdroj obrazu a čas jeho vytvoření. Aplikace zajišťuje veškeré náležitosti, aby mohla být fotografie považována za důvěryhodnou. Zabývá se také získáním informací z obrazu použitím metod zpracování obrazu. Důraz je kladen na bezpečnost aplikace a přenosu dat.

Klientská aplikace v součinnosti se serverovou aplikací kryptograficky ověřuje zdroj a čas vytvoření obrazu. Získává podpis času, digitální podpis, hash, časové razítko od důvěryhodného serveru a GPS souřadnice pro důkaz existence dat před určitým časem. Fotografie může být poté považována za důvěryhodný dokument. Práce je zaměřena na bezpečnou komunikaci aplikace s hardwarem fotoaparátu mobilního zařízení. Důležitá je také bezpečná výměna klíčů, autentizace uživatele, šifrovaná komunikace a zabránění útokům. Prováděna je detekce a rozpoznání objektů v obraze.

Předpokládám, že téměř každý člověk se nacházel v situaci, kdy potřeboval vytvořit dokument prokazující určitou situaci či událost, ke které došlo v daném čase na konkrétním místě. Například když byl svědkem či účastníkem nehody, obětí přestupku, trestného činu, ohrožování nebo omezování. Využití vytvořeného důkazního materiálu může být za účelem prokázání viny, neviny nebo k zabránění další nežádoucí činnosti. Tato práce má význam pro člověka, který se může v budoucnosti nacházet v takové situaci. Umožňuje mu kdykoliv vytvořit dokazující důvěryhodný dokument. V životě občana může ke zmíněným situacím běžně docházet. Řešení této práce může uživateli ušetřit materiální prostředky, zdraví, čas, čest nebo dokonce život.

Mobilní zařízení má pro tento účel velmi výhodné parametry a větší část populace jej má neustále u sebe k dispozici. Často se již mobilní zařízení k tomuto účelu používají. Používají se však samotná bez nějakého systematického řešení, které zajistí právoplatnost vytvořeného důkazu. Takový dokument jako důkazní materiál u různých řízení často selhává. Libovolné mobilní zařízení nelze považovat za důvěryhodný zdroj důkazního materiálu. Tato problematika neobnáší pouze technologické překážky, ale naráží také na různé právní aspekty. Důvěryhodný dokument je z hlediska zákona přesně definovaný a musí splňovat zákonem stanovené podmínky. Samozřejmostí je také plnění technologických a bezpečnostních standardů. Podmínky se netýkají pouze samotného dokumentu, ale celého systému řešícího tuto problematiku. Důležité je zajištění bezpečnosti všech prvků systému a jejich komunikace. Pokud řešení vyžaduje přenos citlivých informací, je nutné zajistit šifrování komunikace. Není optimální ponechat veškerou činnost spojenou se zpracováním obrazu a kryptografickými operacemi na mobilním zařízení. Větší část operací je doporučeno delegovat na server.

Mobilní aplikace nebude díky tomu plýtvat energií, datovými přenosy, operační pamětí a výpočetním výkonem mobilního zařízení. Vhodné kryptografické algoritmy řešící šifrování, hashování, digitální podpis a časové razítko pro tuto aplikaci existují. Důležité je zabezpečit komunikaci s hardwarem mobilního zařízení a zajistit bezpečnou výměnu klíčů, autentizaci a zabránění útokům. Obecné principy pro řešení těchto problémů existují. Použitelných serverových řešení je velké množství a mezi mobilními platformami lze uvažovat jen iOS a Android. Konkurenční řešení, které řeší tuto problematiku komplexně neexistuje. Lze nalézt pouze řešení s velmi omezenou podmnožinou uvažovaných kryptografických operací. Cílem je tedy spojit všechny znalosti a výhody, vyřešit vzniklé problémy, zvolit vhodné postupy a vytvořit kvalitní komplexní řešení, které uživateli pomůže usnadnit život.

Práce navrhuje architekturu pro řešení definované problematiky a přichází s koncepcemi aplikace, které jsou vhodné pro řešení tohoto problému. Jednotlivé koncepce jsou důkladně analyzovány a vzájemně porovnány. Jsou uvažovány výhody a nevýhody jednotlivých představených koncepcí. Výsledkem srovnání je zvolení nejvhodnější koncepce splňující požadovaná kritéria, která bude implementována v rámci diplomové práce. Práce klade důraz kromě tvorby klient-server aplikace s mobilními klienty také na zpracování obrazu, bezpečnost komunikace a kryptografická ověření dokumentu. Výstupem je důvěryhodný dokument opatřený digitálním podpisem, důvěryhodným časovým razítkem a GPS souřadnicemi, ze kterého jsou získány další užitečné informace pomocí zpracování obrazu. Zabývá se taktéž touto problematikou z pohledu práva a využitelností takto získaného důkazního materiálu například u soudního řízení, správního řízení nebo šetření pojišťovny ohledně odpovědnosti vzniklé škody.

Následující druhá kapitola popisuje dosavadní stav. Informuje o vývoji vytváření důvěryhodných dokumentů v historii. Zabývá se problematikou vytváření důvěryhodných dokumentů a představuje vhodné prostředky pro implementaci řešení problematiky. Třetí kapitola popisuje aktuální stav. Je zde detailně popsán konkrétní návrh zvolené koncepce řešení a zadání práce. Definuje také, jakým způsobem se bude hodnotit výsledná práce a kritéria splnění cíle. Čtvrtá kapitola popisuje vlastní práci. Popsána je koncepce díla, práce na klientské a serverové aplikaci, komunikace, datový model, nasazení a fungování. Je ověřena správnost a funkčnost díla. Pátou kapitolu práce tvoří závěr.

Kapitola 2

Shrnutí dosavadního stavu

Tato kapitola práce popisuje dosavadní stav. Dává čtenáři přehled o vývoji v oblasti získávání důvěryhodných dokumentů. Přesně popisuje, jakými způsoby se v dnešní době získávají důvěryhodné digitání dokumenty. Zaměřuje se také na aktuálně dostupná mobilní řešení. Poté na problematiku nahlíží z pohledu návrháře. Je zde popsáno jaké možnosti má vývojář, který se rozhodne k implementaci aplikace řešící tuto problematiku. Začíná od samotné architektury aplikace, zaměřuje se na klientskou a serverovou implementační platformu. Zabývá se dalšími koncepty, které mohou výsledné aplikaci přidat na bezpečnosti a uživatelském komfortu. Nakonec popisuje metody zpracování obrazu pro získání dalších informací z vytvořeného dokumentu. Shrnutí dosavadního stavu je omezeno rozsahově a na skutečnosti vztahující se k diplomové práci.

2.1 Elektronický podpis

Elektronický podpis byl v legislativě České republiky definován Zákonem o elektronickém podpisu a o změně některých dalších zákonů č. 227/2000 Sb.[1], který legislativně definoval elektronický podpis elektronického dokumentu. Tento zákon měl však svá úskalí, kterým se blíže věnuje podkapitola s názvem Legislativní vývoj. Tento zákon byl proto nahrazen novým Zákonem č. 297/2016 Sb. o službách vytvářejících důvěru pro elektronické transakce[5] z důvodu Nařízení Evropského Parlamentu a Rady (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakcena vnitřním trhu a o zrušení směrnice 1999/93/ES[4], který konkrétně definuje jednotlivé pojmy a dává prostor pro národní úpravy. Podle tohoto nařízení je elektronický podpis právně vymezen a definován takto:

> Elektronický podpis - data v elektronické podobě, která jsou připojena k jiným datům v elektronické podobě nebo jsou s nimi logicky spojena a která podepisující osoba používá k podepsání

V České republice se definováním pojmů souvisejích s touto problematikou zabývala ICT Unie, která vydala k této problematice samostatný dokument[22]. Pojem elektronického podpisu definuje následovně:

> Elektronický podpis - Elektronickým podpisem se rozumí údaje v elektronické podobě, které jsou připojené k datové zprávě nebo jsou s ní logicky spojené, a které slouží jako metoda k jednoznačnému ověření identity podepsané osoby ve vztahu k datové zprávě

2.2 Elektronická pečeť a časové razítko

Stejná situace z právního hlediska se týkala také elektronického časového razítka a pečeti neboli značky. Nařízení Evropského Parlamentu a Rady (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakcena vnitřním trhu a o zrušení směrnice 1999/93/ES[4] stanovuje právní rámec pro elektronické pečetě a elektronická časová razítka následovně:

Elektronické časové razítko - data v elektronické podobě, která spojují jiná data v elektronické podobě s určitým okamžikem a prokazují, že tato jiná data existovala v daném okamžiku

Elektronická pečeť - data v elektronické podobě, která jsou připojena k jiným datům v elektronické podobě nebo jsou s nimi logicky spojena s cílem zaručit jejich původ a integritu

ICT Unie tyto pojmy taktéž definuje[22]:

Časové razítko - Kvalifikovaným časovým razítkem se rozumí datová zpráva, kterou vydal kvalifikovaný poskytovatel certifikačních služeb, a která důvěryhodným způsobem spojuje data v elektronické podobě s časovým okamžikem, a zaručuje, že uvedená data v elektronické podobě existovala před daným časovým okamžikem

Elektronická značka - Elektronickou značkou se rozumí údaje v elektronické podobě, které jsou připojené k datové zprávě nebo jsou s ní logicky spojené, a které splňují následující požadavky:

1. jsou jednoznačně spojené s označující osobou a umožňují její identifikaci prostřednictvím kvalifikovaného systémového certifikátu,
2. byly vytvořeny a připojeny k datové zprávě pomocí prostředků pro vytváření elektronických značek, které označující osoba může udržet pod svou výhradní kontrolou,
3. jsou k datové zprávě, ke které se vztahují, připojeny takovým způsobem, že je možné zjistit jakoukoli následnou změnu dat

2.3 Důvěryhodný elektronický dokument

Nařízení Evropského Parlamentu a Rady (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu a o zrušení směrnice 1999/93/ES[4] stanovuje pravidla pro služby vytvářející důvěru. Zejména se jedná o důvěru u elektronických transakcí. Je prvním platným zákonem v České republice, ve kterém je z legislativního hlediska definován pojem důveryhodnosti v oblasti elektronických transakcí. Podrobnosti legislativního vývoje jsou popsány v následující podkapitole. Nařízení definuje pojmy:

Služba vytvářející důvěru - elektronická služba, která je zpravidla poskytována za úplatu a spočívá:

a) ve vytváření, ověřování shody a ověřování platnosti elektronických podpisů, elektronických pečetí nebo elektronických časových razítek, služeb elektronického doporučeného doručování a certifikátů souvisejících s těmito službami nebo

b) ve vytváření, ověřování shody a ověřování platnosti certifikátů pro autentizaci internetových stránek nebo

c) v uchovávání elektronických podpisů, pečetí nebo certifikátů souvisejících s těmito službami

Elektronický dokument - jakýkoli obsah uchovávaný v elektronické podobě, zejména jako text nebo zvuková, vizuální nebo audiovizuální nahrávka

Zákon č. 297/2016 Sb. o službách vytvářejících důvěru pro elektronické transakce[5] výše uvedené nařízení doplňuje. ICT Unie definuje důvěryhodný elektronický dokument a s ním spojené pojmy následovně[22]:

Dokument - Podle §2 písm. e) zákona č.499/2004Sb., o archivnictví a spisové službě a o změně některých zákonů, ve znění pozdějších předpisů každá písemná, obrazová, zvuková nebo jiná zaznamenaná informace, ať již v podobě analogové či digitální, která byla vytvořena původcem nebo byla původci doručena. Dokument má tedy obecnější podobu než písemnost, neboť předpokládá více forem, než pouze písemnou. Podle platné právní úpravy musí být právní úkon podepsán jednající osobou vlastnoručně. V případech, kdy je to obvyklé, může být nahrazen mechanickými prostředky (např. razítkem). Je-li právní úkon učiněn elektronicky, může být podepsán zaručeným elektronickým podpisem.

Elektronický dokument - Elektronickým dokumentem se rozumí dokument v elektronické podobě.

Důvěra - Důvěra je spolehnutí se na něco, očekávání něčeho, je určující faktor v procesu rozhodování. Znamená vztah spoléhání na druhé lidi, instituce nebo věci.

Důvěryhodnost - Důvěryhodnost je vlastnost vztažená k nabízené nebo poskytované službě, ze které je možno odvodit důvěru v řádné provedení této služby.

Digitální dokument je důvěryhodný, pokud jsou splněny následující požadavky:

1. Jedná se o originální (autentický, původní) dokument, nebo jeho odvození z originálního dokumentu (např. stejnopis či jeho konvertovanou verzi)
2. Lze jednoznačně určit původ dokumentu
3. Lze jednoznačně ověřit, že nedošlo k porušení integrity dokumentu
4. V případě kopie, repliky nebo konverze lze doložit shodu s originálem
5. Je zaručena jeho čitelnost
6. Lze jednoznačně prokázat existenci dokumentu v čase

Z důvodu zaručení důvěryhodnosti dokumentu z dlouhodobého hlediska, je nutné využívat komplex služeb důvěryhodných poskytovatelů, jako například ověřování elektronického podpisu/značky, ověřování certifikátů, registr elektronických identit osob (v ČR zatím neexistuje), zachovávání síly kryptografického mechanismu elektronického podpisu/značky a časového razítka, fixace dokumentu formou elektronické značky a/nebo časového razítka a převodu do standardizovaného archivního formátu.

2.4 Elektronické důkazní materiály

Podle Nařízení Evropského Parlamentu a Rady (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakcena vnitřním trhu a o zrušení směrnice 1999/93/ES[4] platí následující tvrzení:

1. Aby se přispělo k obecnému přeshraničnímu využívání služeb vytvářejících důvěru, mělo by být možné použít je ve všech členských státech jako důkaz v soudním a správním řízení. Je na vnitrostátním právu, aby vymezilo právní účinky služeb vytvářejících důvěru, nestanoví-li se v tomto nařízení jinak.

2. Po přiměřenou dobu, i poté, co ukončil svou činnost kvalifikovaného poskytovatele služeb vytvářejících důvěru, eviduje a zpřístupňuje veškeré příslušné informace týkající se dat, která vydal a obdržel, zejména pro účely poskytnutí důkazů v soudním a správním řízení a pro účely zajištění kontinuity služby. Tato evidence může mít elektronickou podobu.

3. Elektronickému podpisu nesmějí být upírány právní účinky a nesmí být odmítán jako důkaz v soudním a správním řízení pouze z toho důvodu, že má elektronickou podobu nebo že nesplňuje požadavky na kvalifikované elektronické podpisy.

4. Elektronické pečeti nesmějí být upírány právní účinky a nesmí být odmítána jako důkaz v soudním a správním řízení pouze z toho důvodu, že má elektronickou podobu nebo že nesplňuje požadavky na kvalifikované elektronické pečetě.

5. Elektronickému časovému razítku nesmějí být upírány právní účinky a nesmí být odmítáno jako důkaz v soudním a správním řízení pouze z toho důvodu, že má elektronickou podobu nebo že nesplňuje požadavky na kvalifikované elektronické časové razítko.

6. Elektronickému dokumentu nesmějí být upírány právní účinky a nesmí být odmítán jako důkaz v soudním a správním řízení pouze z toho důvodu, že má elektronickou podobu.

Podle ICT Unie je důvěryhodný dokument právně nezpochybnitelný[22]. Neměnnost a neporušitelnost dokumentu lze obtížně zaručit, pouze lze činit opatření, které to znesnadňují. Místo toho lze požadovat jednoznačnou detekovatelnost porušení integrity dokumentu. Důkazní materiál důvěryhodného dokumentu by měl obsahovat tyto komponenty:

1. Dokument samotný

2. Základní metadata (sadu základních povinných metadat, která by měla být vždy součástí důkazního materiálu)

3. Záznam všech operací, kterým byl dokument podroben (např. konverze, tisk, export)

4. Informace o provedené konverzi (konverzní doložka), byla-li provedena

5. Elektronický podpis podepisující osoby/el. značka označující organizace či osoby

6. Elektronická značka úložiště a časové razítko dokumentující čas příjmu do úložiště

7. Všechny bezpečnostní prvky (otisky) dokumentu a archivního balíčku

8. Důkazní informace o ověření uznávaného elektronického podpisu/elektronické značky a kvalifikovaných certifikátů

9. Prohlášení o způsobu vkládání, ověřování a uchovávání dokumentů v důvěryhodném úložišti s odkazem na vnitřní politiku organizace/certifikovaný systém

10. Elektronická značka a časové razítko vztahující s k důkaznímu materiálu definující čas generování důkazního materiálu

2.5 Legislativní vývoj

Legislativa České republiky do nedávné doby nedefinovala pojem důvěra pro elektronické transakce resp. ani pojem důvěryhodný dokument. Zákon o archivnictví a spisové službě a o změně některých zákonů č. 499/2004 Sb.[3] definuje pouze pravost dokumentu. Více odpovídající formulace vztahující se k důvěryhodnosti se nachází v Zákoně o dani z přidané hodnoty č. 235/2004 Sb.[2], kde je definována věrohodnost původu dokladu. V žádném zákoně však nebylo možné nalézt informace, jakým způsobem důvěryhodnost dokumentu zajistit. Existoval však Zákon o elektronickém podpisu a o změně některých dalších zákonů č. 277/2000 Sb.[1], který ale neumožňoval jednoznačně identifikovat podepisující osobu pomocí certifikátu. V naší zemi tedy chyběla kvalitní legislativa pro problematiku elektronických podpisů, důvěry elektronických transakcí a autentizaci. Problémem byla především absence vhodného identifikátoru osoby pro elektronickou komunikaci. Situace se ovšem změnila po Nařízením Evropského Parlamentu a Rady (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu a o zrušení směrnice 1999/93/ES[4]. Toto nařízení je také známé pod zkratkou eIDAS. Obecně platí, že nařízení Evropské Unie je právně nadřazeno v případě konfliktu resp. rozporu se zákonem národním, ale nařízení zákony České republiky neruší. Výše zmíněný Zákon č. 277/2000 Sb. nicméně byla Česká republika nucena zrušit a tedy již není platný spolu s dalšími doplňujícími výhláškami. ČR jej nahradila novým Zákonem č. 297/2016 Sb. o službách vytvářejících důvěru pro elektronické transakce[5]. Tento zákon měl být tzv. adaptačním zákonem. Tím bylo myšleno, že se náš zákon adaptuje pro předpis EU a doplní části, které jsou určeny pro národní úpravu. Nový zákon však měl velké zpoždění a po nabytí účinnosti současně platil ještě s nezrušeným Zákonem č. 277/200 Sb. a Nařízením 910/2014. Aktuální doplněné znění Zákona č. 297/2016 Sb. o službách vytvářejících důvěru pro elektronické transakce již ruší starší Zákon č. 277/2000 Sb. V oblasti autentizace došlo k posunu schválením Zákona č. 298/2016 Sb. Zákon, kterým se mění některé zákony v souvislosti s přijetím zákona o službách vytvářejících důvěru pro elektronické transakce, zákon č. 106/1999 Sb., o svobodném přístupu k informacím, ve znění pozdějších předpisů, a zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů[6]. Nabytím platnosti tohoto zákona byla otevřena cesta elektronickým občanským průkazům, které tento zákon schvaluje. Taktéž odstraňuje poplatek za elektronický občanský průkaz a umožňuje ukládat i certifikáty určené pro identifikaci. Průlom z pohledu soukromoprávních poskytovatelů identit přinese pravděpodobně Zákon č. 250/2017 Sb. o elektronické identifikaci[7]. Prokazování totožnosti je využíváno i u soukromých subjektů, například bank a podobně. Tudíž se předpokládá poskytování identit i pro soukromoprávní subjekty. Elektronické občanské průkazy se začnou vydávat nejdříve po nabytí platnosti Zákona č. 250/2017 Sb.

2.6 Standardy pro elektronický podpis, časové razítko a bezpečnost přenosu dat

Aby byl elektronický dokument považován za důvěryhodný musí být opatřen elektronickým podpisem. Pro formát elektronického podpisu existuje standard CMS definovaný aktualizovaným RFC 5083[25] a především jeho rozšířená verze CAdES, o kterém informuje RFC 5126[36]. CAdES definuje European Telecommunications Standards Institute (ETSI) v dokumentu ETSI TS 101 733[15]. Tato rozšířená verze je taktéž definována jako standard European Telecommunications Standards Institute (ETSI)[9] vzhledem k zmíněnému Nařízení Evropského Parlamentu a Rady (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu a o zrušení směrnice 1999/93/ES[4] dále eIDAS. Standardem jsou také považovány známé formáty elektronických podpisů XAdES, PAdES a také kontejner podpisových formátů ASiC[16]. Pro elektronický podpis ve shodě s CAdES[15] a nařízením eIDAS[4] se používají kryptografické algoritmy RSA a DSA. Algoritmus RSA byl vyvinut trojicí L. Rivest, A. Shamir a L. Adleman, podle které byl tento algoritmus i pojmenován. Záznamy o algoritmu pochází z roku 1977 a definován byl v článku s názvem A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, který byl vydán v roce 1978[38]. Digital Signature Algorithm (DSA) je součástí standardu Digital Signature Standard (DSS) americké vlády, který byl navržen americkým institutem National Institute of Standards and Technology (NIST) a definován v dokumentu FIPS 186[32]. Společně s algoritmem pro digitální podpis se využívá kryptografická hashovací funkce.

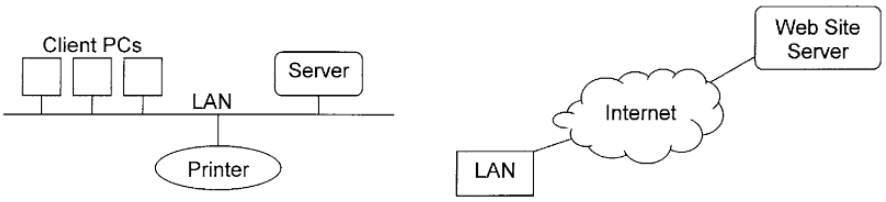
Pro zajištění důvěryhodnosti elektronického dokumentu je nutností jej opatřit elektronickým časovým razítkem. Pro elektronické časové razítko je používán především standard TSP, který je definován RFC 3161[8], který je aktualizován novějším ESSCertIDv2 definovaným v RFC 5816[40].

Podmínkou pro zachování důvěryhodnosti elektronického dokumentu odpovídajícího výše zmíněným zákonům a standardům je také zajištění integrity. To znamená nutnost zabezpečit přenášená data vhodným a především bezpečným kryptografickým algoritmem. Pomocí bezpečného kryptografického algoritmu v případě modifikace dat detekujeme změnu v přenášeném elektronickém dokumentu a ten již nadále není považován za důvěryhodný. Obecně je jednoduché detekovat modifikaci v přenášených datech, ale mnohem obtížnější je modifikaci zamezit. Vhodným kryptografickým nástrojem pro zajištění integrity jsou kryptografické hashovací algoritmy. Pokud jsou přenášeny citlivé údaje, je nutné použít kryptografický šifrovací algoritmus pro zabezpečení přenášených dat.

2.7 Architektura řešení

Architektura klient-server je velmi výhodná pro implementaci softwarových řešení tohoto typu. Klient-server model je nejběžnější a nejpoužívanější forma síťové architektury, která je používána pro datovou komunikaci[21]. Její popularita rostla s expandováním World Wide Webu. Název této architektury pochází ze spojení dvou spolupracujících prvků systému a to klienta a serveru. Klient je definován jako systém nebo program, který požaduje funkci nebo službu jednoho nebo více dalších systémů či programů nazývaných servery pro provedení specifického úkolu[21]. Server je systém nebo program, který po obdržení žádosti od jednoho nebo více klientských systémů či programů poskytuje požadovanou funkci, službu nebo prostředky[21]. Obvykle klient a server se nacházejí na odděleném hardwaru a

komunikují přes počítačovou síť. Klient, který odesílá požadavek na server je iniciátorem komunikace. Server splní požadavek a odesílá klientovi odpověď. Obecně služba serveru je abstrakce počítačových prostředků. Klient nemá přehled jak server zpracovává požadovaný úkol, pouze rozumí jeho odpovědi. To znamená, že klient a server spolu komunikují pomocí předem stanoveného komunikačního protokolu, který přesně definuje obsah a formát dat požadované služby. Klient a server si obvykle vyměnují zprávy pomocí komunikačního vzoru požadavek-odpověď[24]. Klient pošle požadavek a server vrací odpověď. Aby si spolu klient a server rozuměli, musí komunikovat pomocí společného jazyka a musí dodržovat pravidla definované komunikačním protokolem. Klient-server komunikační protokoly komunikují na aplikační vrstvě TCP/IP modelu, která definuje základní vzory dialogu[14]. Pro formalizování datové výměny může server implementovat rozhraní pro programování aplikací tzv. API (Application Programming Interface)[10]. API je vrstva abstrakce pro přístup ke službě. Použití API je velmi výhodné, protože server tak může obdržet mnoho požadavků od mnoha různých klientů v krátkém čase. Server může provádět pouze omezený počet úkolů v daný čas a je pouze na plánovacím systému serveru prioritizovat přijímané požadavky od klientů pro zpracování. Pro zajištění proti zneužití a maximalizování dostupnosti server může omezovat dostupnost klientům.



(a) Lokální architektura (b) Architektura se vzdáleným serverem

Obrázek 2.1: Klient-server architektura[21]

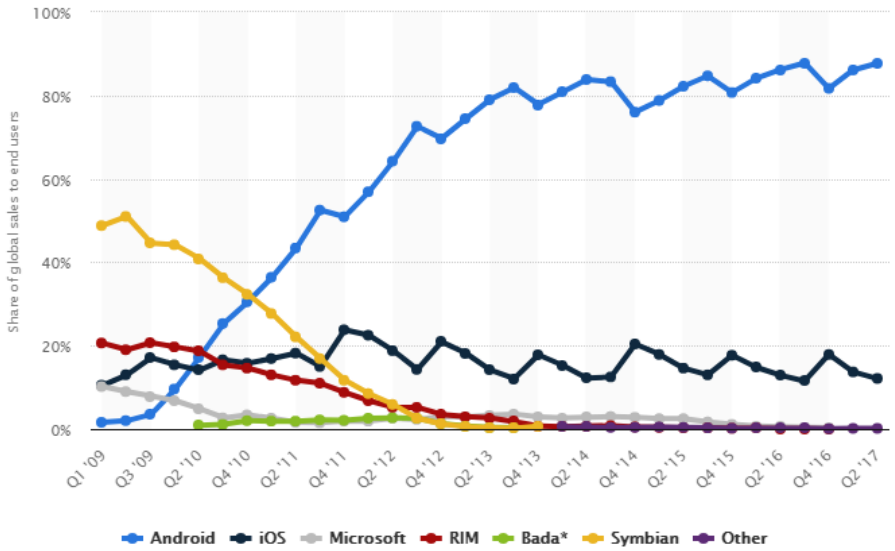
Historie klient-server modelu se datuje od 70. let 19. století. Předchůdcem tohoto modelu byla procedura RJE (Remote Job Entry) představená v rámci operačního systému OS/360 od IBM v roce 1964[26]. Požadavek byl úkol (job) a odpovědí byl výstup (output). Ve Standfordském výzkumném institutu počítačoví vědci vytvářející ARPANET definovali pojmy user-host (using host) a server-host (serving host), které byly užívány v dokumentech RFC4[41] a RFC5[39]. Jazyk DEL (Decode Encode Language) definovaný v RFC5 byl vytvořen pro přijímání kódovaných příkazů z počítače (user-host), kterému vracel hlášení stavu v internetových paketech. Server-host přijímá pakety s požadavkem, dekóduje požadavek a vrací user-hostu formátovaná data. Poprvé se pojem klient vyskytuje v dokumentu výzkumného centa společnosti Xerox v Palu Altu, kde autoři J. Israel, J. Mitchell a E. Sturgis pečlivě definují pojem klient a popisují rozdíl mezi pojmy klient (client) a uživatel (user)[27]. Pod pojmem host je považován počítač připojený k síti. Nicméně za pojmy klient a server se může skrývat také počítačový program.

Budoucností klient-server architektury jsou aplikační servery, které umožňují využít veškeré výhody internetové technologie[21]. Aplikační server je hlavním prvkem modelu a nachází se mezi klienty a zdrojem dat. Stará se správu a běh aplikací. Umožňuje vývoj a nasazení komplexních webových aplikací. Zajišťuje spolehlivost, škálovatelnost a snadnou ovladatelnost. Aplikační server může být integrován s webovým serverem. Využití tohoto modelu umožňuje usnadnit práci a vyřešit technické problémy při vývoji webové aplikace.

2.8 Platformy klientské aplikace

Z hlediska klientské části aplikace zadání práce požaduje řešení s využitím mobilních technologií. S ohledem na situaci na trhu s mobilními zařízeními resp. s ohledem na zastoupení jednotlivých platforem u těchto zařízení je celkem omezený výběr mobilních platforem vhodných pro vývoj klientské části aplikace. Pro vývoj klientské části aplikace jsou vhodné pouze ty platformy, které zaujímají na trhu významný podíl. Předpoklad je uvedení aplikace pro nejširší možné spektrum uživatelů. Před samotným vývojem aplikace je proto vhodné analyzovat statistiky prodeje mobilních zařízení koncovým zákazníkům pro dané platformy mobilních operačních systémů. Důležitým údajem pro výběr platformy je také celkový podíl platforem u zákazníků.

Podle serveru statista.com ve druhém kvartálu roku 2017 drží prvenství v prodeji koncovým zákazníkům s podílem na trhu 87.7% platforma Android. Na druhém místě s podílem na trhu 12.1% se umístila platforma iOS. Ostatní platformy utržily zbylých 0.2% podílu na trhu[43]. Podle serveru statcounter.com je ve druhém kvartálu celkový podíl u zákazníků platformy Android mírně nižší a to konkrétně 72.5%. Druhé místo v podílu na trhu opět zaujímá platforma iOS s 19.5%. Na ostatní platformy zbývá 8%[42]. Podle trendu ostatní platformy (všechny kromě iOS a Android) neustále a dlouhodobě (od roku 2009) ztrácí podíl na trhu. Méně než 1% zákazníků si aktuálně koupí mobilní zařízení s jiným mobilním operačním systémem než je iOS nebo Android. Z toho vyplývá, že není vhodné pro tyto platformy vyvýjet aplikace za účelem masivního rozšíření či zisku. Většinou se jedná o zanikající platformy, jejichž vývoj se zastavil a nové verze jsou v nedohlednu. Tyto platformy významným způsobem přišly a nadále přichází o klíčové vývojáře a vývojářská studia. Celkově tedy připadají v úvahu pouze dvě největší platformy, které jsou nejčetněji zastoupeny u světové populace. Jsou to iOS od společnosti Apple a Android od společnosti Google. Pro ostatní existující platformy nemá z hlediska jejich zastoupení na trhu a aktuálního trendu smysl vyvýjet aplikace, které mají být určeny pro velké množství lidí nebo mají generovat zisk.



Obrázek 2.2: Podíl na trhu mobilních operačních systémů[43]

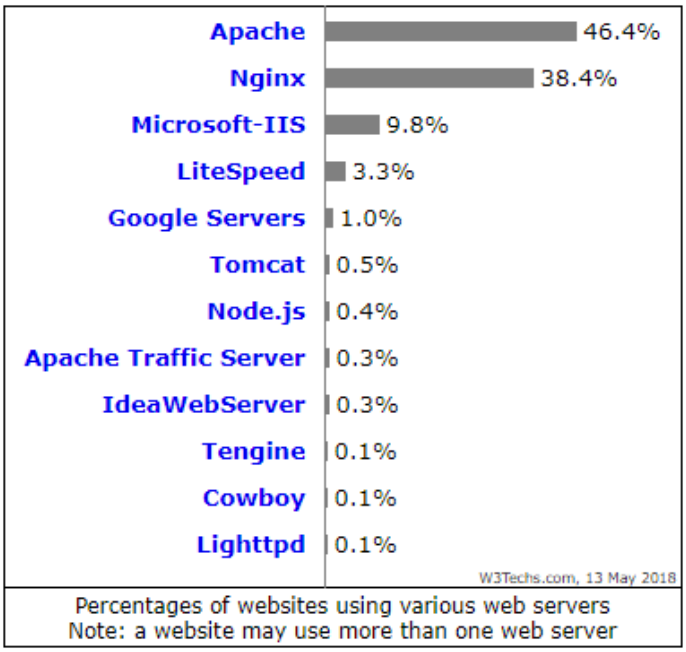
Důležité je před výběrem vhodné mobilní platformy resp. platforem zvážit multiplatformní vývoj, pro nějž existuje řada nástrojů. Většinou se však jedná o komerční a často placená řešení, která nejsou nutně potřebná. Mezi multiplatformní nástroje pro vývoj patří Appery.io, který je vhodný spíše pro tvorbu business aplikací. Je to platforma založená na cloudovém řešení s využitím vizualních vývojových nástrojů a integrovaných back-end služeb. Přichází se spoustou šablon a také podporuje responzivní webové aplikace. Používají jej například AT&T, ESPN a Samsung. Pro multiplatformní vývoj je vhodný také nástroj Appcelerator. Je vhodný pro uživatele, kteří mají zkušenosti s vývojem v JavaScriptu a webových technologiích. Nabízí také integrované cloud služby, knihovny a různá rozšíření dostupná v modulu marketplace. Dalším multiplatformním řešením je užití nástroje Xamarin, který spadá pod Microsoft. Je vhodný pro vývojáře, kteří s oblibou používají C#. Umožňuje vytvářet aplikace pro iOS, Android a Windows. Xamarin také poskytuje interaktivní dashboard zobrazující data v reálném čase. Uživatel má tedy přehled o uživatelích a sezeních. K dispozici jsou také užitečné metriky. Není vhodný pro komplexní a složitější aplikace. Dále existují také multiplatformní nástroje GeneXus, Kinvey, Appzillon a další. Tento typ vývoje není však vhodný v každé situaci a pro všechny aplikace.

V této práci se zaměřím na vývoj klientské části aplikace pro jednu konkrétní platformu. K vývoji mobilní aplikace čistě pro mobilní operační systém iOS se vyplatí použít nativní vývojové prostředí Apple Xcode. Prostředí je vytvořeno přímo společností Apple. Obsahuje spoustu pluginů a umožňuje přizpůsobení podle představ uživatele. Jazyky pro vývoj jsou Swift a Objective-C. Pro vývoj čistě Android aplikace je také výhodné využít nativní prostředí Android Studio přímo od společnosti Google. Vývojovým jazykem u tohoto nástroje je Java nebo Kotlin.

2.9 Platformy serverové aplikace

Vhodných serverových řešení pro implementaci této části práce je velká spousta. Na server nejsou kladena žádná významná implementační omezení. Důležité je, aby byl server schopný komunikovat s klientskou aplikací. Server musí také komunikovat s dalším důvěryhodným serverem pro zajištění časového razítka. Je kladen důraz na bezpečnost komunikace a použití vhodných kryptografických algoritmů. To je však problematika komunikace samotné a není ve většině případů příliš závislá na konkrétním serverovém řešení. Komunikace bude tedy podrobněji řešena v následující podkapitole. Požadavkem na server je však spouštění nástrojů a aplikací běžících na stroji serveru pro zpracování obrazu a kryptografické operace. U serverové aplikace nemusí vývojář sledovat, která platforma je nejvíce používána a zastoupena. Důležité je brát v potaz kvalitu, bezpečnost, stabilitu a dostupnost výsledného řešení. Samozřejmě při implementaci serverové části aplikace je výhodnější použít oblíbené a rozšířené nástroje a technologie z důvodu snadnějšího řešení problémů, kvalitnější technické podpory, velkého množství materiálů a rozsáhlé diskuse uživatelů. U většiny volených technologií je potřebný webový server pro zpřístupnění aplikace a zpracování komunikace pomocí HTTP protokolu. Komunikace tohoto typu je popsána v následující podkapitole.

Nejvhodnější je použití webového serveru Apache, který je nejrozšířenějším webserverem a používá jej k 13.5.2018 podle serveru w3techs.com 46.4% webů[45]. Na druhém místě se nachází webserver Nginx, který používá 38.4% webů. Třetí příčka patří IIS od společnosti Microsoft, který používá 9.8% webových stránek a aplikací. Ostatní webservery zaujímají zbylých 5.4%. Zajímavostí je, že webserver Tomcat používá pouze 0.5% a Node.js 0.4% webů. Použití daného webserveru je také závislé na konkrétní zvolené implementační platformě serverové aplikace.

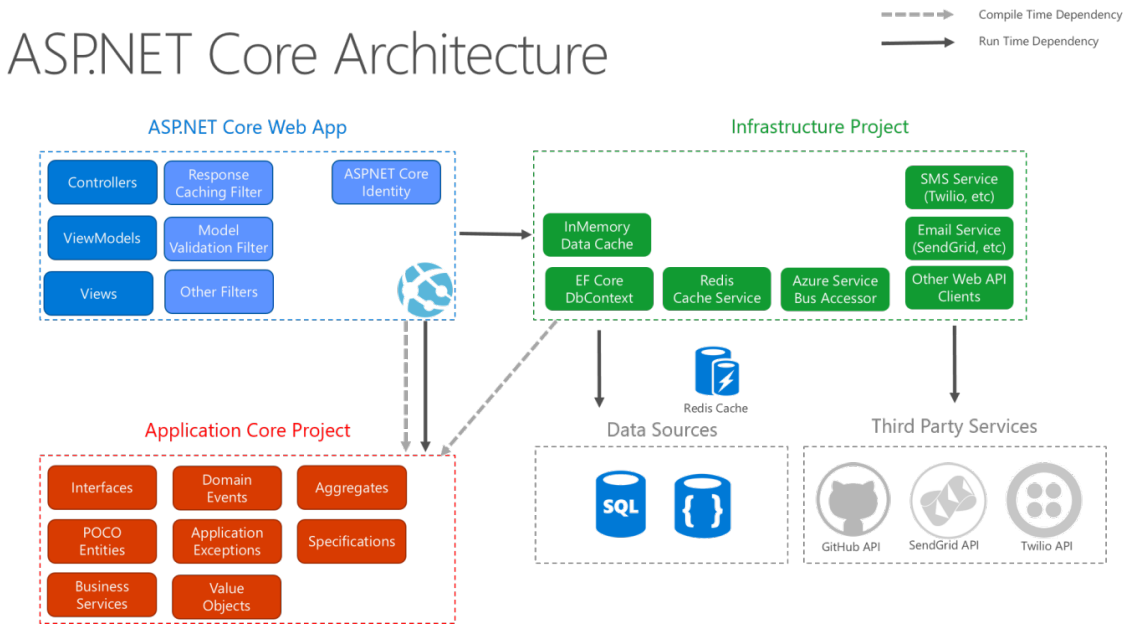


Obrázek 2.3: Použití webových serverů[45]

Nejsnadnější a nejrychlejší způsob implementace serverové aplikace je pomocí technologie PHP. Pro zprovoznění serverové aplikace je potřeba webserver. V tomto případě je vhodný například Apache. Práci nám může usnadnit velké množství existujících frameworků. Tyto frameworky umožňují rychlý vývoj, dobře organizovat a udržovat kód, zbavit se některých starostí s bezpečností, dodržovat MVC vzor a používat moderní postupy při vývoji. Mezi nejpoužívanější PHP frameworky patří Laravel, který má velký ekosystém s platformou pro okamžité nasazení. K dispozici je mnoho návodů, šablonový engine, autentizace, relace, řazení do front, cachování a RESTful API. Dalším významným PHP frameworkem je Symfony. Používá jej Drupal, phpBB a také Laravel. Má rozsáhlou vývojářskou komunitu. Dostupné jsou opětovně využitelné knihovny pro vytváření formulářů, konfiguraci objektů, směrování, autentizaci, šablony a další. Zavedeným PHP frameworkem je CodeIgniter. Vyžaduje minimum konfigurací a tím šetří spoustu času. Je založený na architektuře MVC. Další zajímavé PHP frameworky jsou například Yii2, Nette a Slim. Existuje však mnoho dalších. Pro ukládání dat může být například použit databázový systém MySQL.

Pokročilejší implementaci serverové aplikace lze realizovat pomocí technologií od společnosti Microsoft. Webové aplikace v .NET Frameworku či .NET Core jsou implementovány v programovacím jazyce C#. Webové stránky pak mohou být založeny na HTML5, CSS a JavaScriptu. Pomocí této technologie lze vytvářet moderní aplikace, služby a webové stránky. Vývoj je poměrně jednoduchý a rychlý. Výsledný produkt může být určen pro miliony uživatelů[28]. Výhodou je, že se jedná o velmi oblíbenou platformu, kterou využívá velké množství vývojářů. Podporuje také oblíbený návrhový vzor MVC. Pro implementaci aplikací pomocí této technologie je vhodné nativní prostředí Visual Studio od společnosti Microsoft. Pro nasazení vytvořené aplikace je nutné na Microsoft Serveru nakonfigurovat webserver IIS a Web.config. Potřeba je nakopírovat vytvořenou aplikaci a data, nainstalovat bezpečnostní certifikáty, nastavit registry a případně provést další potřebné úkoly[29]. Pro snadný deployment pak prostředí přímo nabízí funkcionalitu pro nasazení vytvořené apli-

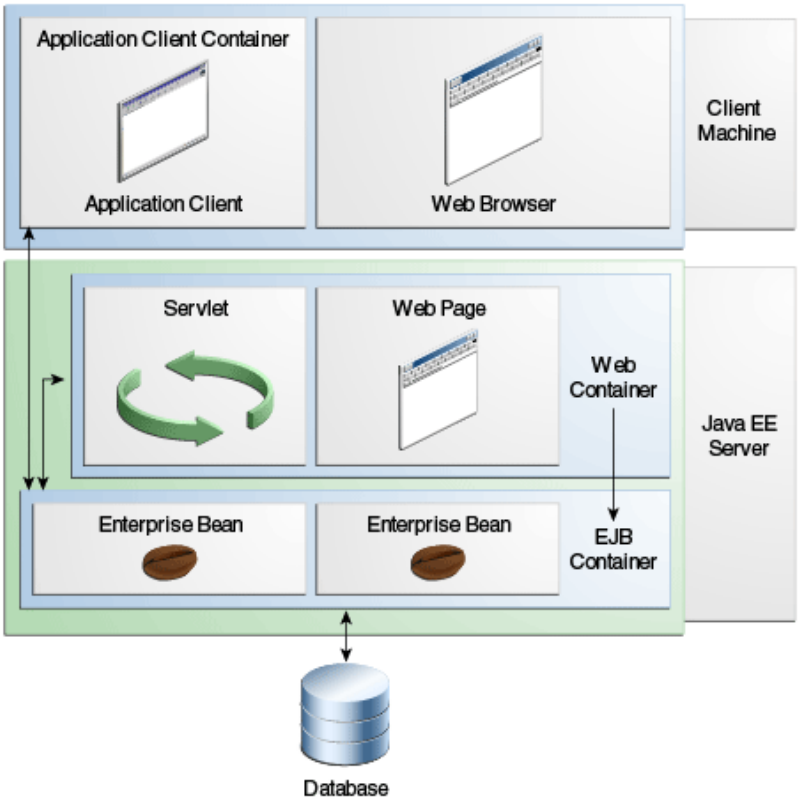
kace na vybraný cílový server. Tato funkcionalita umožňuje přimo spouštět i databázové skripty. Může být zvolen například databázový systém SQL Server od společnosti Microsoft.



Obrázek 2.4: ASP.NET Core architektura[30]

Alternativou pokročilejší serverové platformy pro implementaci serverových aplikací je platforma Java EE neboli Java Enterprise Edition. Aplikace pro tuto platformu jsou implementovány v jazyce Java. Pro front-end webových aplikací a webové stránky lze využít HTML5, CSS a JavaScript. Technologie Java EE se skládá z několika komponent. Nástrojem pro tvorbu webových aplikací je Java Servlet, který funguje na principu zpracování HTTP požadavků a generování HTML stránek. Pro vytváření dynamických HTML webových stránek s použitím Java kódu slouží technologie Java Server Pages (JSP). Grafické uživatelské rozhraní a aplikační logiku webové aplikace lze snadno rozdělit pomocí technologie Java Server Faces (JSF), která definuje interface pomocí speciálních XML tagů, kterým se data předávají pro zobrazení a editaci ze standardních Java Beanů. Pro snadný a rychlý vývoj existuje mnoho JSF frameworků, například PrimeFaces, BootsFaces a další. Vkládání závislostí mezi jednotlivými komponentami programu zajišťuje technologie Dependency Injection (DI). Jedna komponenta může používat druhou, aniž by na ni měla v době sestavování programu referenci. Pro přístup k relačním databázím slouží technologie Java Persistence API (JPA). Serverové komponenty umožňující modulární tvorbu podnikových aplikací se nazývají Enterprise Java Beans (EJB). Vychází z klasických Java Beanů a výhodou je znovu-použitelnost kódu, oddělení business logiky od prezentační vrstvy aplikace, transakční zpracování, bezpečnost, jednodušší testování a integrace. Přístup k legacy systémům zajišťuje technologie Java Connector Architecture (JCA) a pro zasílání zpráv Java Messaging Services (JMS). Pro implementaci Java EE webových aplikací je vhodné použít prostředí NetBeans od Oracle Corporation, IntelliJ IDEA od JetBrains nebo Eclipse od Eclipse Foundation. Výhodou je, že Java EE webové aplikace nejsou závislé na konkrétním operačním systému serveru. Pro nasazení je potřeba zvolit a nakonfigurovat aplikační server. Nejznámější aplikační servery jsou Tomcat od Apache, GlassFish od Sun a JBoss od RedHat nově přejmenovaný na WildFly. Pro zvýšení bezpečnosti řešení se v některých

případech používá současně s aplikačním serverem také klasický webový server. Nejčastěji je používán webový server Apache nebo Nginx. Vhodné je použít například databázový systém MySQL.

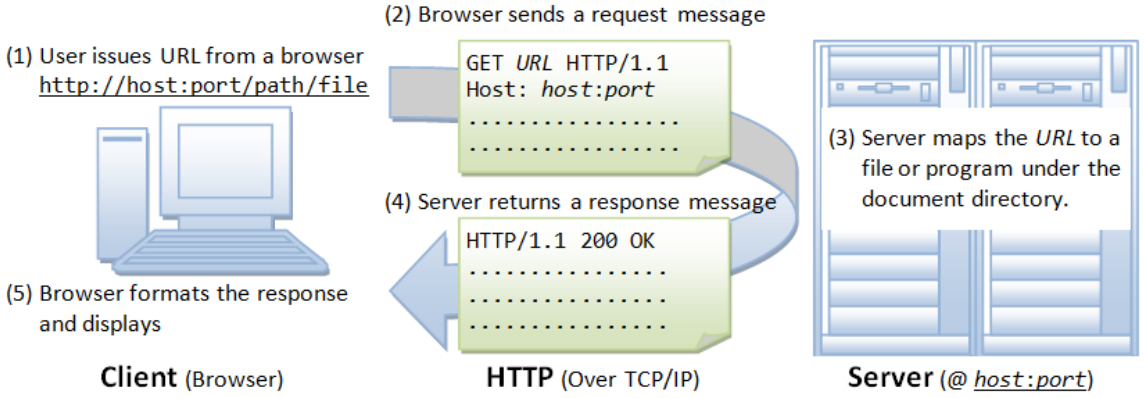


Obrázek 2.5: Java EE architektura[34]

2.10 Komunikace klientské a serverové aplikace

Pro výměnu zpráv mezi klientem a serverem je vhodné použít request-response model komunikace. Příkladem takové komunikace může být HTTP, RPC, RMI a další. V této práci se zaměřím na nejčastěji používané protokoly. Komunikace klientské a serverové aplikace realizováná pomocí HTTP protokolu je velmi podobná jako komunikace webového prohlížeče a serveru poskytujícího webové stránky. Tento způsob komunikace se serverem je vhodný pro svoji jednoduchost, univerzálnost a použití pro všechny možné klientské a serverové platformy. S daným sereverem pak může komunikovat webová aplikace, desktopová aplikace i mobilní aplikace a všechny tyto aplikace mohou běžet na libovolných platformách a mohou být vytvořeny pomocí libovolných technologií. World wide web je založen na třech hlavních technologiích a to URL, HTTP a HTML. Standard URL byl definován v RFC3986[11] a aktualizován RFC6874[13] a RFC7230[19]. HTTP protokol byl nejprve definován RFC2068[17]. Později byl aktualizován RFC2616[18], který byl dále aktualizovaný několika dalšími dokumenty (RFC2817, RFC5785, RFC6266 a RFC6585). Zmíněné RFC dokumenty už jsou ovšem zastaralé a nahrazené novějšími návrhy standardů (RFC7230[19], RFC7231[20], RFC7232, RFC7233, RFC7234, RFC7235). Pro programování distribuovaných aplikací s využitím těchto technologií je nutné jim detailně porozumnět. URL iden-

tifikuje nějaký zdroj, například produkt, uživatele, domovskou stránku a podobně. HTTP komunikace je založena na odesílání požadavku o konkrétní zdroj klientem na server. Server poté vrací odpověď klientovi na jeho požadavek. Odpověď může být dokument, binární soubor a další. Tuto odpověď nazýváme reprezentací zdroje. Je nutné zachovat adresovatelnost, tzn. každý zdroj by měl mít svoji unikátní URL. HTTP požadavek klient odesílá přes internet na webserver, který jej zpracuje. Server poté odešle odpověď klientovi.



Obrázek 2.6: HTTP požadavek a odpověď[23]

Číslo na začátku odpovědi značí kód odpovědi. Je to jednoduchý a rychlý způsob, jak server sdělí klientovi výsledek jeho požadavku. Nejčastěji je vracen výsledek 200 OK. To znamená, že požadavek byl splněn bez problémů. Kompletní seznam všech kódů je dostupný v RFC7231[20]. Klient přijme a zpracuje odpověď od serveru. V případě webového prohlížeče ji graficky zobrazí uživateli. Tato stránka však nemusí být jediná. Získáním odpovědí se uživateli postupně může odhalit celá struktura zdrojů. Kromě GET metody HTTP požadavku existují ještě metody HEAD, POST, PUT, DELETE, CONNECT, OPTIONS a TRACE definované v RFC7231[20]. HTTP sezení jsou krátká a server neví nic o stavu aplikace klienta. Klient nemá přímou kontrolu nad stavem zdrojů. Web funguje takto flexibilně díky principům a architektuře Representational state transfer (REST). Stav aplikace je uchováván u klienta a server jím může manipulovat zasíláním reprezentací. Stav zdroje je uchováván na serveru, ale klient jím může manipulovat zasláním reprezentace. V minulosti vznikla celá řada různých protokolů. Většina z nich zanikla kvůli své složitosti, nepoužitelnosti, nemožnosti upravení na míru požadavkům atd. Přeživší protokoly uměly něco, co HTTP protokol neumí (FTP, BitTorrent, SSH, atd). HTTP je totiž velmi dobrý protokol a většinou nám na vše stačí metoda GET[37]. Pro jednoduchost použití World Wide Webu při tvorbě distribuované aplikace je vhodné vytvořit RESTful API. Komunikace funguje stejným způsobem, jen odpověď není HTML dokument. Odpověď je jednoduchá, stručná a je v datovém formátu, který je srozumitelný a zpracovatelný klientem. Obvykle se používá datový formát JSON definovaný v RFC4627. Formát je definován v HTTP hlavičce v poli Content-Type.

Komunikace klienta a serveru může být realizována také pomocí technologie vzdáleného volání procedur zvané Remote Procedure Call (RPC). Poprvé jej implementovali A. D. Birrell a B. J. Nelson z výzkumného centra Xerox Palo Alto Research Center[12]. Oblíbená byla implementace Open Network Computing Remote Procedure Call (ONC RPC) od společnosti Sun Microsystems na platformě Unix definována v RFC1057[44]. Existuje také implementace RPC od společnosti Microsoft zvaná Microsoft Remote Procedure Call

(MSRPC), která je založena na DCE/RPC[31]. V rámci frameworku .NET lze využít aplikaci Windows Communication Foundation (WCF) poskytující RPC. Existuje velká spousta dalších implementací a některé umožňují také komunikovat pomocí HTTP protokolu. Komunikace probíhá pomocí zasílání RPC zpráv. Technologie se nestará o to, jakým způsobem je zpráva doručena. Zabývá se pouze specifikací a interpretací zprávy. Nicméně aplikace může požadovat informace nebo kontrolu nad transportní vrstvou prostřednictvím rozhraní. RPC komunikace může být implementována na různých transportních protokolech. Pokud komunikace probíhá na TCP, tak transportní vrstva odvede vetšinu práce související s přenosem. Je-li použit UDP protokol, aplikace musí zajistit vlastní time-out, opětovné odesílání a detekci duplicit. Tyto služby RPC vrstva neposkytuje. Protokol RPC zpráv je definován pomocí jazyka eXternal Data Representation (XDR). Komunikace probíhá tak, že client první posílá zprávu (volání) na server a čeká na jeho odpověď. Volání (call) obsahuje parametry procedury. Odpověď pak obsahuje výsledek procedury. Jakmile je odpověď serveru přijata, tak klient extrahuje výsledek procedury a jeho činnost je ukončena. Na straně serveru se vyčkává na příchod zprávy (volání). Když dorazí zpráva, tak z ní server extrahuje parametry procedury, provádí výpočet a odesílá výsledek v odpovídající zprávě. Poté opět čeká na přijetí dalších zpráv (volání). RPC technologie podporuje autentizaci. Mohou být použity různé autentizační protokoly. Speciální políčko v RPC hlavičce indikuje použitý protokol. Pro autentizaci může být použita Unix autentizace nebo DES.

Alternativou k RPC je metoda invokace metod zvaná Remote Method Invocation (RMI), která je především používaná na platformě Java[35]. Je vhodná pro tvorbu distribuovaných systémů. Nevýhodou RPC je, že nefunguje dobře na distribuovaných objektových systémech, kde je potřeba komunikace mezi objekty programu nacházející se na odlišných adresách v paměti[33]. RMI je navrženo pro práci v aplikačním prostředí Javy a díky homogenímu prostředí virtuálního stroje využívá kdekoliv výhod objektového modelu platformy Java. Komunikace se vzdálenými objekty pomocí RMI vypadá podobně jako standardní invokace metody. RMI se skládá ze tří vrstev. První vrstvu tvoří stub a skeleton. Stub se nachází na straně klienta mezi aplikací klienta a remote reference. Stub je něco jako zástupce vzdáleného objektu. Po zavolání metody vzdáleného objektu u klienta stub vytváří blok informací, který obsahuje id vzdáleného objektu, číslo metody a parametry zabalené do marshall streamu. Skeleton se nachází na straně serveru. Rozbaluje marshall stream a volá metodu objektu. Skeleton poté získá návratovou hodnotu a zabalí ji do marshall streamu. V novějších verzích Javy se místo skeletonu používá RMI překladač rmic. Návratovou hodnotu rozbaluje a vrací stub. Druhou vrstvou je Remote reference, která je prostředníkem mezi stub a skeleton. Třetí vrstva je transport, která zajišťuje spojení mezi dvěma virtuálním stroji (JVM). RMI využívá TCP protokol nad kterým používá ještě Java Remote Method Protocol (JRMP). Pro získávání a poskytování objektů se používají RMI registry. Na tyto registry server umístí stub, který si potom může klient stáhnout, aby získal referenci na vzdálený objekt. Vzdálený objekt lze také získat jako parametr nebo návratovou hodnotu voláním metody.

2.11 Zpracování obrazu, detekce a rozpoznávání objektů v obraze

Použité metody při zpracování obrazu budou závislé na konkrétní použité implementaci detektoru a rozpoznávání. Některé detektory již provádí předzpracování vstupní fotografie. Obecně se před detekcí obvykle provádí změnšování a zvětšování fotografie (scale). Dále

lze taktéž aplikovat různé afinní transformace. Například zkosení (shear), rotace (rotation), aplikace různých filtrů, prahování (thresholding) a podobně. Detektory mohou být založeny na algoritmu Adaboost nebo Waldboost. Pro implementaci detektoru lze taktéž použít neuronové sítě.

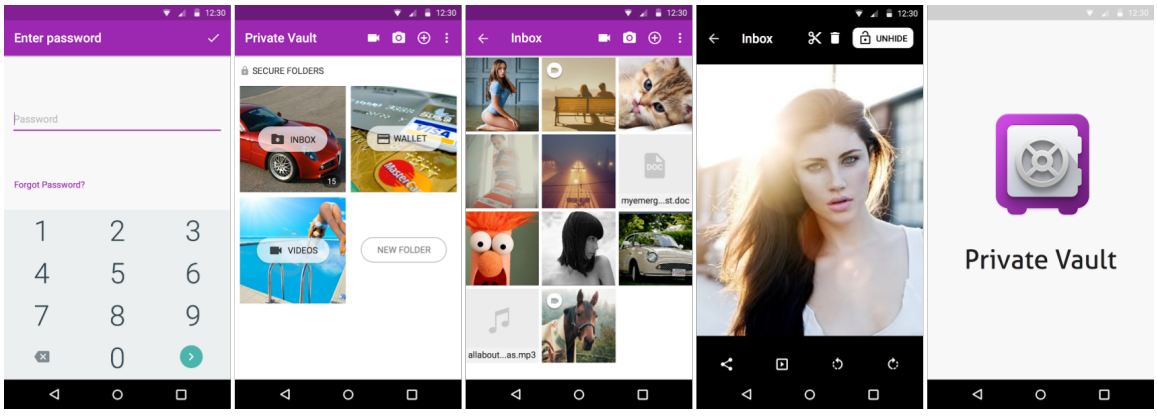
Po provedení detekce je vhodné provést zpracování obrazu před samotným rozpoznáním. Vhodné je provést ořez dektekovaných oblastí fotografie, narovnání obrazu a provést segmentaci. Nejběžnějším typem segmentace je prahování (thresholding). Dále se pro segmentaci využívají metody založené na regionech. Mezi tyto metody patří spojování oblastí, štěpení oblastí, štěpení a spojování, watershed a shluková analýza. Segmentace rozdělí obraz na jednotlivé oblasti zájmu. U těchto oblastí pak definujeme charakteristiky reprezentující objekt. Mezi charakteristiky patří barva, tvar, konvexnost, konkávnost, plnost a momentové příznaky. Poté jsou objekty klasifikovány do tříd na základě charakteristiky.

2.12 Existující řešení

Obdobné komplexní řešení problematiky, kterým se zabývá tato práce doposud neexistuje. Především žádná existující aplikace neřeší současně všechny uvedené problémy. Prvním problémem je bezpečná komunikace aplikace s hardwarem fotoaparátu mobilního zařízení a zabránění podvrhnutí komunikace útočníkem. Druhým problémem je bezpečná výměna klíčů před komunikací, šifrovaná komunikace a autentizace uživatele. Třetím problémem je zábránění útokům, snahám o ovládnutí aplikace a pokusům o znepřístupnění aplikace útočníkem. Čtvrtým problémem v pořadí je zaručení integrity. Posledním problémem je zajištění všech náležitostí důvěryhodného dokumentu, především získání digitálního podpisu, hashe a časového razítka. Existující aplikace většinou řeší pouze omezenou podmnožinu funkcionalit uvažovaných v navrhovaném řešení, které je popsáno v následující kapitole.

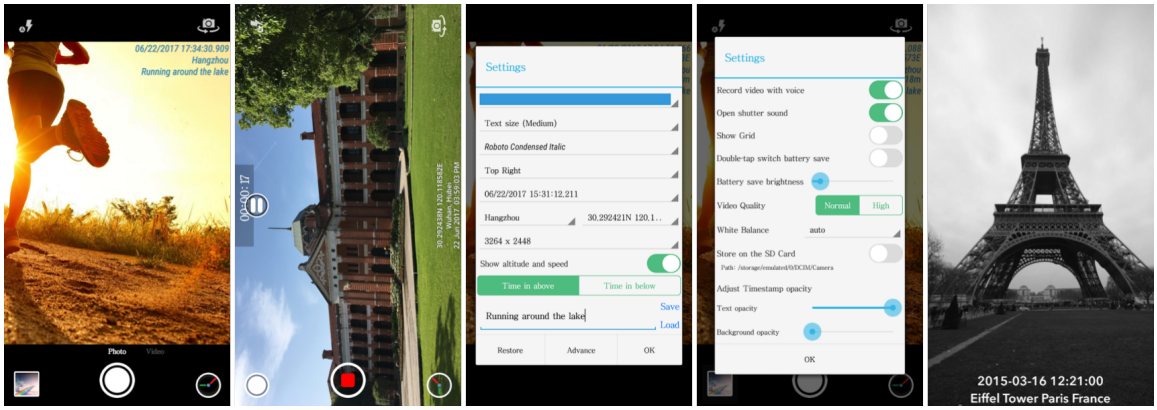
Pro bezpečné pořizování fotografií, provádění kryptografických operací a bezpečné ukládání na platformě Android jsou dostupné následující aplikace.

Aplikace Hide Pictures & Videos - VAULT slouží k bezpečnému ukládání a šifrování dokumentů. Je zaměřena na jednoduchost. Umožňuje šifrovat různé typy dokumentů. Používá šifrovací algoritmus AES 256. Data jsou dostupná pouze po autentizaci. Aplikace provádí automatické odhlášení po určité době nečinnosti. Používá nativní fotografickou aplikaci. Lze provádět jednoduché úpravy fotografií.



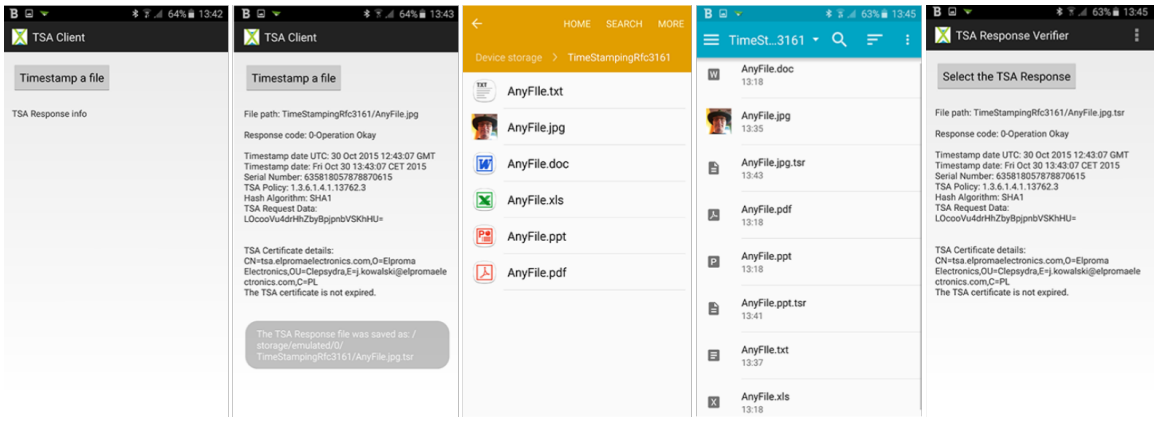
Obrázek 2.7: Hide Pictures & Videos - VAULT

Aplikace Timestamp Camera Free vytváří časové razítko pro fotografie a videa. Přidává do fotografie a videa také vodoznak s časovým razítkem. Podporuje mnoho formátů časového razítka. Přidává GPS souřadnice a adresu lokace. Podporuje změnu rozlišení.



Obrázek 2.8: Timestamp Camera Free

Aplikace RFC3161 TimeStamping Client získává časové razítko od serveru Elproma na adrese https://tsa.elpromaelectronics.com. Podporuje různé typy dokumentů. Aplikace RFC3161 TimeStamping Verifier provádí ověření časového razítka.



Obrázek 2.9: RCF3161 Timestamping Client a RFC3161 Timestamping Verifier

Existují další aplikace poskytující obdobnou funkcionalitu. Například Safe Camera - Photo Encryption, Keepsafe Photo Vault, Safe Photo Free a další aplikace pro bezpečné ukládání dat. K získání časového razítka pro dokumenty lze použít například OriginStamp - Trusted timestamping.

Kapitola 3

Zhodnocení současného stavu a plán práce

V této kapitole je zhodnocen současný stav práce. Detailně jsou popsány výhody a nevýhody různých řešení. Zaměřuje se na nedostatky a nevyřešené problémy, které je vhodné nebo nutné vyřešit pro dosažení kvalitního řešení na základě znalosti stavu. Je zde analyzována dostupnost a množství potřebných zdrojů pro běh implementovaného řešení. Kapitola se zabývá také funkcionalitou řešení. Jsou popsány vstupy, výstupy, parametry a použité prostředky pro realizaci řešení. Definuje kritéria hodnocení řešení, které jednoznačně stanoví hodnocení navrženého řešení vzhledem k zadání práce. Umožňuje tedy zhodnotit, zda je splněno zadání práce a dosaženo požadovaného cíle. Na závěr kapitoly je detailně popsáno navržené řešení a zadání práce.

3.1 Zhodnocení současného stavu klient-server aplikací

Pozitivně lze hodnotit dostupné a velmi rozvinuté technologické prostředky a nástroje, které tvoří kvalitní základ pro praktické řešení problematiky. Pro implementaci řešení práce lze využít velké množství potenciálních platforem. Je možné vybírat podle výrobce softwaru, programovacího jazyka, bezpečnosti implementace, osobních preferencí, uživatelských požadavků, řešené problematiky a dalších aspektů. Při výběru dané platformy lze často i zvolit implementační programovací jazyk a frameworky pro usnadnění práce. Výhodou je existence velkého množství bezplatných open source platforem, jazyků, prostředí, nástrojů a algoritmů, které jsou realně použitelné a nasaditelné pro naplnění potřeb velkého množství uživatelů. Pozitivním předpokladem je disponibilita Evropské unie a České republiky kvalitní legislativou, která řeší problematiku elektronického podpisu, časové známky, časového razítka a důvěryhodného dokumentu.

Co je nedostatečné, co chybí nebo co lze hodnotit negativně, tak absenci digitálního identifikátoru každého občana a digitálního klíče pro elektronický podpis. Více informací o legislativním vývoji je k dispozici v druhé kapitole práce. Elektronický občanský průkaz je teprve ve fázi testovaní vybraným vzorkem uživatelů. Prozatím však můžeme chybějící klíče nahradit jiným řešením. Na mobilních platformách dále není vyřešeno zabezpečení fotografického čipu proti podvrhnutí obrazu při převzetí kontroly útočníkem nad operačním systémem. Není také spolehlivě a bezpečně vyřešena problematika výměny a uchovávání klíčů mezi klientskou a serverovou aplikací a současně autentizace uživatele se zabráněním

potenciálním útokům. Velmi obtížně jsme schopni také detekovat podvrhy před fotoaparátem mobilního zařízení.

Z hlediska parametrů výsledného řešení jsem se snažil nalézt kompromis mezi cenou a kvalitou. Především jsem se zaměřil na kvalitní open source technologická řešení nezávislá na operačním systému. Jediný náklad tvoří provoz virtuálního privátního serveru. Provoz VPS je však méně nákladný a výhodnější než provoz vlastního serveru. Řešení hostovaného VPS serveru má nespornou výhodu lepší datové konektivity a napojení na páteřní optické sítě včetně záložního síťového připojení. Kromě energetické výhody z hlediska spotřeby je ještě velmi důležitý záložní zdroj energie pro případ výpadku. Datová centra bývají připravena na veškerý typ rizik spojených s přírodními živly i lidským faktorem. Podmínky, jakými disponují datová centra, se s omezeným rozpočtem v domácích podmínkách velmi obtížně vytvářejí. Existuje velké množství poskytovatelů hostingových služeb. Vhodný je výběr dle dostupných technologií, kvality, dostupnosti, rychlosti, spolehlivosti, latence, podpory, ceny, osobních preferencí a případně dalších parametrů. Další nákladovou položku mohou tvořit také využívané služby pro digitální podpisy, časová razítka a certifikáty. Snaha je zvolit levné nebo bezplatné řešení. Nativní technologie pro tvorbu klientské mobilní aplikace jsou bezplatné. Lze nalézt také bezplatné technologie pro multiplatformní vývoj.

Operační systém	Ubuntu Server 16.04 LTS
CPU	1 vlákno Intel Xeon
RAM	1GB
Typ úložiště	SSD
Velikost úložiště	20GB
Přenos	2 TB/měsíc
Hypervizor	VMware
SLA	99.80%

Tabulka 3.1: Parametry serveru

Důraz je kladen na dostupnost serverové a klientské aplikace. Serverová aplikace bude umístěna na vzdáleném hostovaném virtuálním serveru VPS s dostupností přes 99%. Klientská aplikace bude dostupná v obchodě zvolené platformy. Její dostupnost bude také velmi blízko 100%.

Pro provoz serverové a klientské aplikace je důležité stanovit potřebný výpočetní výkon a různá omezení s tím související. Virtuální privátní server VPS disponuje 1GB operační paměti RAM, využívá solid state disk SSD a pohání jej výpočetní procesorová jednotka Intel Xeon s vyhrazeným vláknem. Toto řešení bylo otestováno při relativně nízké zátěži. V případě reálně vyšších nároků na hardware virtuálního serveru je díky technologii VPS velmi snadné alokovat další výpočetní či paměťové prostředky. Z důvodu výkonosti a bezpečnosti je pro běh mobilní klientské aplikace vyžadována u operačního systému určitá mininimální verze.

3.2 Popis funkcionality systému

Funkcionalita je postupně popsána nejprve z pohledu klienta a poté z pohledu serveru. Pro zabránění úniku citlivých dat při komunikaci klientské a serverové aplikace je komunikace

šifrována. Nejprve je však nutné zajistit bezpečnou výměnu klíčů, autentizaci uživatele a zabránění potenciálních útoků neautorizovaného útočníka.

Funkcionalita klientské aplikace Klientská aplikace běžící na mobilním zařízení řeší autentizaci uživatele. Nový uživatel má možnost se registrovat a tím vytvořit svůj uživatelský účet. Stávající uživatel, který má již vytvořený účet, se může přihlásit. Registrace a přihlášení se provádí v součinnosti se serverovou aplikací. Po autentizaci uživatele proběhne počáteční komunikace klienta a serveru, při které server stanoví počáteční nastavení pro zpracování obrazu a kryptografické operace. Jakmile klient obdrží a zpracuje požadovanou konfiguraci, aplikace je připravena k dalšímu použití. Mobilní aplikace uživateli zobrazuje galerii fotografií. Seznam fotografií poskytuje server. Pro každý prvek v seznamu fotografií jsou dostupné operace pro zobrazení původní fotografie, zobrazení podrobností fotografie, verifikaci kryptografických prvků, detekci objektů v obraze, rozpoznání objektů v obraze a mazání fotografií. Uživatel má možnost pořídit fotografii pomocí fotoaparátu mobilního zařízení. Před vytvořením fotografie je získáno časové razítko od zařízení, aby mohla být dokázána platnost skutečností i před vytvořením fotografie. Aplikace zajišťuje bezpečnou komunikaci s hardwarem fotoaparátu mobilního zařízení. Obraz není podvržen útočníkem. Pořízená fotografie je poté podepsána pomocí algoritmu digitálního podpisu a zpracována dle požadavků stanovených serverem. Z původní fotografie je poté vytvořen hash. Jsou získány GPS souřadnice polohy zařízení a je vytvořena miniatura fotografie. Časové razítko ze zařízení, digitální podpis, hash, GPS souřadnice a miniatura fotografie jsou poté bezpečným způsobem přeneseny na server a uloženy v databázi. Pro fotografie je nutné zajistit kvalitní úložiště. Z galerie fotografií lze zobrazit původní fotografii a její podrobnosti. Fotografie je zobrazena ze zařízení a podrobnosti fotografie jsou získány ze serveru. Klientská aplikace umožňuje prostřednictvím serveru provádět verifikaci kryptografických prvků fotografie. Server provede ověření jednotlivých prvků a vrací výsledek klientovi. Pomocí serveru klientská aplikace také zajišťuje detekci a rozpoznání objektů na fotografii. Server vrací klientovi rozpoznanou registrační značku vozidla. Dostupný je také uživatelský účet a jeho správa. Aplikace zobrazuje počáteční nastavení od serveru a spravuje klíče.

Funkcionalita serverové aplikace Server zajišťuje registraci a autentizaci uživatele. Po autentizaci server stanovuje v počáteční komunikaci s klientem podmínky pro zpracování obrazu a kryptografické operace. Server přijímá od klienta fotografie. Po zpracování požadavku ukládá časové razítko ze zařízení, podpis, hash, GPS souřadnice a miniaturu fotografie do databáze. Původní fotografii umisťuje do úložiště. Server poté získává časové razítko hashe původní fotografie od důvěryhodného serveru. Přijatý hash původní fotografie server odesílá v žádosti na důvěryhodný server a obdrží časové razítko, které uchová v databázi. Aplikace poskytuje klientovi podrobnosti fotografie. Server umožňuje voláním požadavku provádět verifikaci kryptografických prvků fotografie, detekci a rozpoznání registračních značek na fotografii. Server poskytuje informace o uživateli a provádí správu uživatelského účtu.

Nástroje a vývojová prostředí Klientská aplikace bude vyvýjena na platformě Android v prostředí Android Studio. Testování aplikace bude prováděno především na mobilním telefonu Google Pixel s Android 8.1 Oreo. V rámci Android aplikace bude využito Camera2 API, Retrofit2 klient pro komunikaci se serverovou aplikací pomocí vytvořeného API a Gson pro konverzi mezi JSON a POJO objekty. Server bude implemen-

tován na platformě Java EE v prostředí NetBeans. Aplikace poběží na VPS s operačním systémem Ubuntu Server 16.04 LTS. Na serveru bude nainstalován webový server Apache, aplikační server Tomcat, databáze MySQL a nástroj phpMyAdmin. Pro implementaci serverové aplikace bude použit JAX-RS a Jersey 2 pro tvorbu RESTful API, JDBC pro spojení s databází, JPA pro objektově relační mapování, JSP pro dynamické stránky a JSF pro uživatelské rozhraní. Bezpečná komunikace bude zajištěna pomocí HTTPS protokolu. Pro výměnu klíčů se bude používat algoritmus RSA. Pro šifrování bude zvolen nejvhodnější algoritmus podle priority. Hash fotografie se bude vytvářet pomocí algoritmu SHA-256. Digitální podpis v souladu s legislativními požadavky bude využívat RSA.

Ověření a hodnocení Kritéria hodnocení řešení jsou rozdělena na ověření klientské aplikace, serverové aplikace, komunikačního API a kryptografických náležitostí fotografie jako důvěryhodného dokumentu. Klientská aplikace bude testována stažením z Google Play. Budou postupně ověřeny jednotlivé případy užití definované v návrhu. Testovat se budou jednotlivé vstupy a výstupy zda odpovídají stanoveným požadavkům. Serverovou aplikaci lze testovat i samostatně. S použitím například nástroje Postman lze snadno vytvářet HTTPS požadavky a přijímat odpovědi. Tímto způsobem lze testovat samotné API i výstupy serverové aplikace. Porovnáním získaných a definovaných vstupů a výstupů tak můžeme ověřit správnost serverové aplikace. Další kritérium tvoří hodnocení všech potřebných náležitostí fotografie z pohledu důvěryhodnosti dokumentu. Ověřuje se, zda digitální podpis, hash a časové razítko odpovídá bezpečnostním normám, technologickým standardům a legislativním požadavkům. Důležitá je také kontrola neporušení integrity. Celkové řešení musí být také testováno z hlediska bezpečnosti simulovanými útoky, pokusy o získání citlivých dat, pokusy o získání kontroly nad systémem nebo pokusy o znepřístupnění služby ostatním uživatelům. Pro ověření aplikace bude na závěr proveden uživatelský experiment.

3.3 Návrh klientské a serverové aplikace

Návrh řešení je rozdělen z pohledu klientské a serverové aplikace. Klientská mobilní aplikace bude implementována na platformě Android. Serverová aplikace bude implementována na platformě Java EE.

Návrh klientské aplikace Verze operačního systému Android bude omezena na minimální verzi API level 21. Jedná se o Android verze 5.0 tzv. Lollipop. Komunikace mezi klientskou a serverou aplikací bude realizována pomocí HTTPS protokolu přes RESTful API vytvořené na straně serveru. V rámci klientské aplikace bude implementovaný RESTful API klient Retrofit2, který bude komunikovat se serverovou aplikací. RESTful API klient bude využívat Gson pro konverzi objektů. Přenášená data budou ve formátu JSON a komunikace bude šifrovaná. Výměna klíčů bude realizována pomocí SSL/TLS algoritmem RSA. Šifrování bude mít také na starost SSL/TLS, který zvolí vhodný podporovaný algoritmus dle priority. TLS/SSL bude používat certifikát od Let's Encrypt. Pro zabránění útokům bude omezen počet neúspěšných pokusů o autentizaci. Poté bude uživatel vyzván k vyčkání časového intervalu pro další autentizační pokusy. Bude omezen počet registrací z konkrétní adresy pro daný den. Omezen bude také počet požadavků pro každého uživatele za určený časový interval. Bezpečná komunikace s hardwarem fotoaparátu na platformě Android bude implementována pomocí Android Camera2 API. Toto API umožňuje přímo komunikovat

s hardwarem fotoaparátu. Aplikace pomocí API používaný fotoaparát jako zdroj blokuje a žádná jiná aplikace jej ve stejnou chvíli nemůže použít. Náhled fotoaparátu bude při fotografování zobrazen uživateli na obrazovce mobilního zařízení. Uživatel může provést vizuální kontrolu fotografie a odhalit podvrhnutí útočníkem. V době psaní práce nebyla známa žádná bezpečnostní hrozba, která by umožňovala útočníkovi ovládnout nebo podvrhnout komunikaci s hardwarem přes toto API. Pro digitální podpis vytvořené fotografie bude používán algoritmus RSA. Digitální podpis bude odpovídat technologickým standardům a legislativním požadavkům. GPS souřadnice budou získávány z lokalizačních dat mobilního zařízení. Polohu lze získat od GPS jednotky nebo pomocí síťové infrastruktury. Hash fotografie bude vytvořen pomocí algoritmu SHA-256.

Návrh serverové aplikace Aplikace bude využívat webserver Apache, který stojí před aplikačním serverem Tomcat od Apache Software Foundation. Webový server bude zajišťovat veškerou komunikaci s klienty a požadavky bude předávat aplikačnímu serveru. Apache bude sloužit pro zajištění bezpečné komunikace, ale také bude chránit aplikační server před útoky a bude zodpovědný za filtrování nevhodných požadavků. Tím zajistí bezpečnost a odolnost aplikačního serveru a dostupnost aplikace. Komunikace klienta s webovým serverem bude šifrována pomocí HTTPS protokolu na standardním portu. Webový server bude používat pro komunikaci s aplikačním serverem JServ protokol. Aplikační server nebude dostupný zvenčí, ale pouze na loopbacku serveru. Na serveru bude pro komunikaci s klienty vytvořeno RESTful API pomocí technologií JAX-RS a Jersey 2. Pro správu uživatelských účtů a pro ukládání dat uživatelů se bude používat databázový systém MySQL. Snadná práce s databázovým systémem a správa uživatelských dat bude zajištěna pomocí nástroje phpMyAdmin. Správa systému bude dostupná ve webové aplikaci na serveru. Webová aplikace bude umožňovat prohlížení uživatelských dat. V rámci zpracování fotografie bude před detekcí a rozpoznáním objektů v obraze prováděno zvětšování, změnšování, rotace, zkosení a další transformace. Pro detekci a rozpoznání registračních značek v obraze bude používán JavaANPR. Serverová aplikace pro získání časového razítka kontaktuje důvěryhodný server poskytující tuto službu. Komunikace probíhá pomocí HTTPS protokolu. Server také bude umožňovat verifikaci všech kryptografických prvků fotografie. Pro časové razítko lze využít následující servery:

```
tsa.postsignum.cz:444/TSS/HttpTspServer
freetsa.org
time.centrum.pl
dse200.ncipher.com/TSS/HttpTspServer
tsa.safecreative.org
zeitstempel.dfn.de
tsa.tecxoft.com
timestamp.comodoca.com/rfc3161
sha256timestamp.ws.symantec.com/sha256/timestamp
timestamp.geotrust.com/tsa
timestamp.globalsign.com/scripts/timestamp.dll
ca.signfiles.com/tsa/get.aspx
services.globaltrustfinder.com/adss/tsa
tsp.iaik.tugraz.at/tsp/TspRequest
```

3.4 Zadání práce

Klient Nejprve je plánována práce na klientské Android aplikaci. Připraví se jednotlivé obrazovky a logika. Vytvoří se registrace, autentizace, zaslání zapomenutého hesla a vysunovací navigační menu. Menu bude dostupné v hlavních aktivitách každého modulu aplikace. Do menu se doplní dostupné moduly aplikace a odhlášení. Nejprve se vytvoří modul Fotografie. V modulu se vypracuje seznam pořízených fotografií, fotografování, zobrazení fotografie, podrobnosti fotografie, verifikace kryptografických prvků fotografie, detekce objektů v obraze, rozpoznání objektů v obraze a mazání fotografií. Implementováno bude razítkování, podepisování, hashování, získání GPS polohy, vytvoření miniatury fotografie a zpracování obrazu. Následně se bude pracovat na modulu Uživatel. Vytvořen bude profil uživatele a úprava uživatelských informací. Implementováno bude vytvoření a mazání miniatury uživatelské profilové fotografie. Poté bude vytvořen modul Nastavení, kde bude zobrazeno nastavení od serveru z počáteční komunikace. Klient bude spravovat klíče v modulu Klíče. Implementován bude také RESTful API klient a volání požadavků.

Server Práce se poté přesune na stranu serverové aplikace. Nejprve bude zajištěn virtuální privátní server s operačním systémem Ubuntu. Na serveru bude provedeno počáteční nastavení a bude nainstalován potřebný software. Bude nainstalován aplikační server Tomcat a webový server Apache. Následně bude nastavena bezpečná komunikace webového serveru s klientskou aplikací. Webový server bude zabezpečen pomocí HTTPS. Pro ukládání dat bude nainstalován databázový systém MySQL, který bude následně zabezpečen. Na serverové aplikaci na platformě Java EE bude vytvořeno zabezpečené RESTful API. Serverová aplikace bude napojena na databázi. Poté bude implementována autentizace a registrace. V rámci webové aplikace bude vytvořena správa systému a zobrazení uživatelských dat. Vytvoří se seznam fotografií, zobrazení fotografie, podrobnosti fotografie, verifikace kryptografických prvků fotografie, detekce objektů v obraze, rozpoznání objektů v obraze, mazání fotografie a úprava uživatelských informací.

Práce na některých funkcionalitách aplikace bude probíhat paralelně na klientské a serverové aplikaci. Především v případě komunikace bude vytvořen požadavek na serveru a volání požadavku na klientovi pro získání dat pro určitou funkcionalitu aplikace.

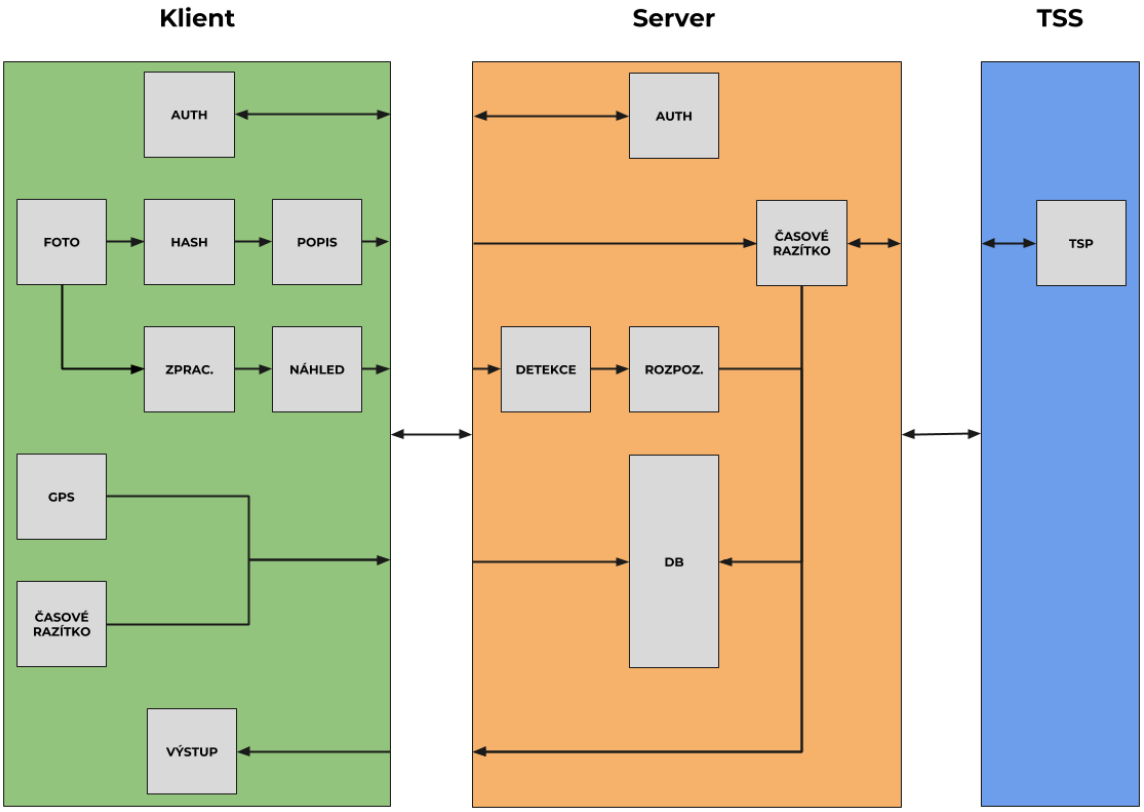
Kapitola 4

Popis vlastní práce

Kapitola popisuje podstatu a koncepci diplomové práce. Zaměřuje se na způsob vytvoření práce, prostředky potřebné k dosažení cíle a výsledky této práce. Popsány jsou zde rysy řešení a rozhodnutí, která k řešení vedla. V této části je také popsána funkcionalita, fungování a použití práce. Tato kapitola neslouží jako uživatelský návod k použití, ten je uveden v příloze práce. Obsahuje nezbytné náležitosti pro pochopení formy práce, struktury klientské a serverové aplikace, datového modelu a komunikace. Tato kapitola se bude na rozdíl od návodu zabývat technickým pohledem na práci. Součástí tohoto popisu bude i nasazení a zprovoznění výstupních aplikací v praxi. Závěrem kapitoly je ověřena správnost a funkčnost díla.

4.1 Popis základní koncepce díla

Práci tvoří systém skládající se ze dvou vzájemně interagujících komponent, tj. klientská a serverová aplikace. Klientská mobilní aplikace na platformě Android provádí bezpečné pořízení fotografie pomocí fotoaparátu zařízení. Je zajištěna přímá komunikace s hardwarem zařízení pomocí Android Camera2 API, blokování fotoaparátu dalším procesům v zařízení a zabránění podvrhům na softwarové či hardwarové úrovni mobilního zařízení. Po vytvoření fotografie je v mobilním zařízení získáno časové razítko. Dále je fotografie podepsána algoritmem SHA256withRSA pomocí soukromého RSA klíče uživatele. Podpis je uložen v mobilním zařízení a ve vytvářeném datovém objektu fotografie. Následně je vytvořen SHA-256 hash fotografie, který je uložen ve vytvářeném objektu. Fotografie je uložena v mobilním zařízení a pro zobrazení náhledu fotografie je vytvořena miniatura fotografie. Miniatura je ukládána do objektu fotografie. Objekt fotografie, který je naplněn daty a původní fotografie je přenesena na server pomocí jediného požadavku. Serverová aplikace implementována na platformě Java EE přijímá fotografii a datový objekt fotografie. Doplňuje do objektu umístění fotografie na serveru. Fotografie je uložena přímo v adresářové struktuře serveru. Server po obdržení fotografie zajišťuje časové razítko od důvěryhodného serveru. Na tento server zasílá požadavek s SHA-256 hashem fotografie a obdrží token a časové razítko. Takto získaná data přidává serverová aplikace do vytvářeného objektu fotografie, který poté ukládá do databáze. Klient umožňuje zobrazit seznam vytvořených fotografií. V kontextové nabídce každé fotografie má uživatel možnost zobrazit fotografii, zobrazit podrobnosti fotografie, verifikovat kryptografické prvky fotografie (podpis, hash, časové razítko), detekovat a rozpoznat registrační značku vozidla na fotografii. Samozřejmostí je možnost smazat fotografii, která smaže fotografii ze zařízení včetně podpisu, fotografii na serveru a objekt v databázi.



Obrázek 4.1: Blokové schéma

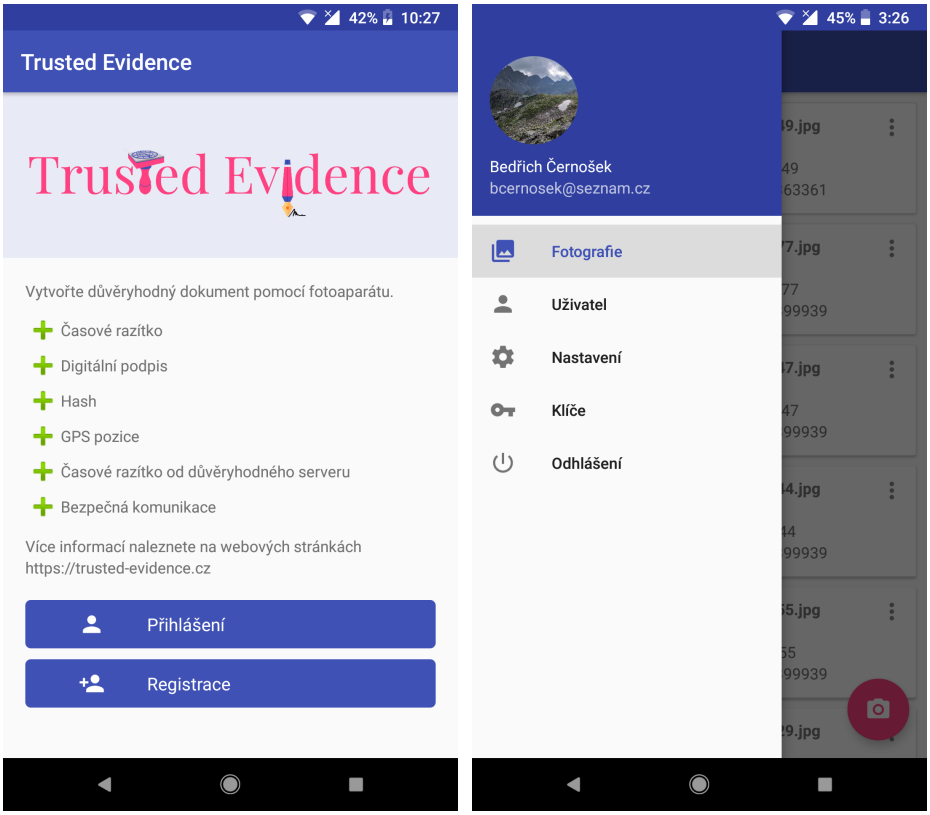
Součástí klientské aplikace je správa uživatelského účtu. Uživatel v tomto modulu aplikace má možnost zobrazit a upravit uživatelské informace. V nabídce menu klientské aplikace je také možnost zobrazení počátečního nastavení od serveru, které nelze měnit. Klientská aplikace umožňuje také spravovat klíče. Součástí serverové webové aplikace je správa systému. Uživatel v rámci webové aplikace může zobrazit seznam fotografií, prohlížet fotografii, zobrazit podrobnosti fotografie, verifikovat kryptografické prvky fotografie, detekovat, rozpoznat registrační značku vozidla a mazat fotografii. Administrátor systému může navíc provádět ve webové aplikaci správu uživatelských účtů. Uživatel může provádět pouze úpravu svého vlastního účtu. Vytvořený systém je obohacen o další koncepty. Komunikace mezi klientskou a serverovou aplikací je implementována pomocí RESTful API. Je zajištěna bezpečná výměna klíčů při ustanovení komunikace. Komunikace je šifrována pomocí HTTPS. Pro ukládání dat slouží databázový systém MySQL s přístupem přes connection pool. Je zabráněno útokům na webovém a aplikačním serveru. Aplikace umožňuje vytvářet pomocí fotoaparátu zařízení důvěryhodný dokument, který je použitelným důkazním materiálem pro soudní či správní řízení.

4.2 Popis práce na klientské aplikaci

Pro vývoj klientské aplikace na platformě Android bylo použito na počátku vývoje aktuální Android Studio ve verzi 3.0.1. Android studio bylo v průběhu práce postupně aktualizováno

na novější verze. Vytvořen byl nový projekt s prázdnou hlavní aktivitou, podporou C++ a omezením na minimální verzi API level 21 tj. Android 5.0, který je označován jako Lollipop.

Hlavní obrazovka Hlavní MainActivity obsahuje logo aplikace, základní informace o aplikaci a dvě tlačítka. Informace obsahují popis, kde v jedné větě je popsána funkcionalita aplikace a následně jsou uvedeny základní parametry aplikace. Jedno z níže umístěných tlačítek slouží k přihlášení do aplikace a druhé k registraci v rámci aplikace. Vzhled hlavní aktivity je definován v activity_main layoutu.



(a) Hlavní obrazovka (b) Menu

Obrázek 4.2: Hlavní obrazovka a menu

Přihlášení Autentizace existujícího uživatele je prováděna v LoginActivity, která obsahuje formulář pro zadání přihlašovacího jména a hesla. Vzhled login aktivity je definován v activity_login layoutu. Formulář umožňuje uživateli pamatovat si přihlašovací údaje pro budoucí přihlášení přepnutím switch přepínače. Uživatel se přihlašuje stiskem tlačítka Přihlásit. Přihlašovací údaje jsou po stisku tlačítka ověřeny voláním požadavku na server. Pod tímto tlačítkem je ještě uživateli nabízena možnost zapomenutého hesla. Funkcionalita zapomenutého hesla se nachází v LostPasswordActivity, kde je formulář pro vyplnění e-mailu uživatele. Stiskem tlačítka Odeslat je odeslán požadavek na server, odkud jsou odeslány na zadaný e-mail přihlašovací údaje. Podmínkou je, že uživatel se zadaným e-mailem existuje. Funkcionalita zaslání přihlašovacích údajů e-mailem je popsána v podkapitole Popis práce na serverové aplikaci. V případě úspěšné autentizace je uživateli vrácen objekt počátečního nastavení a je spuštěna

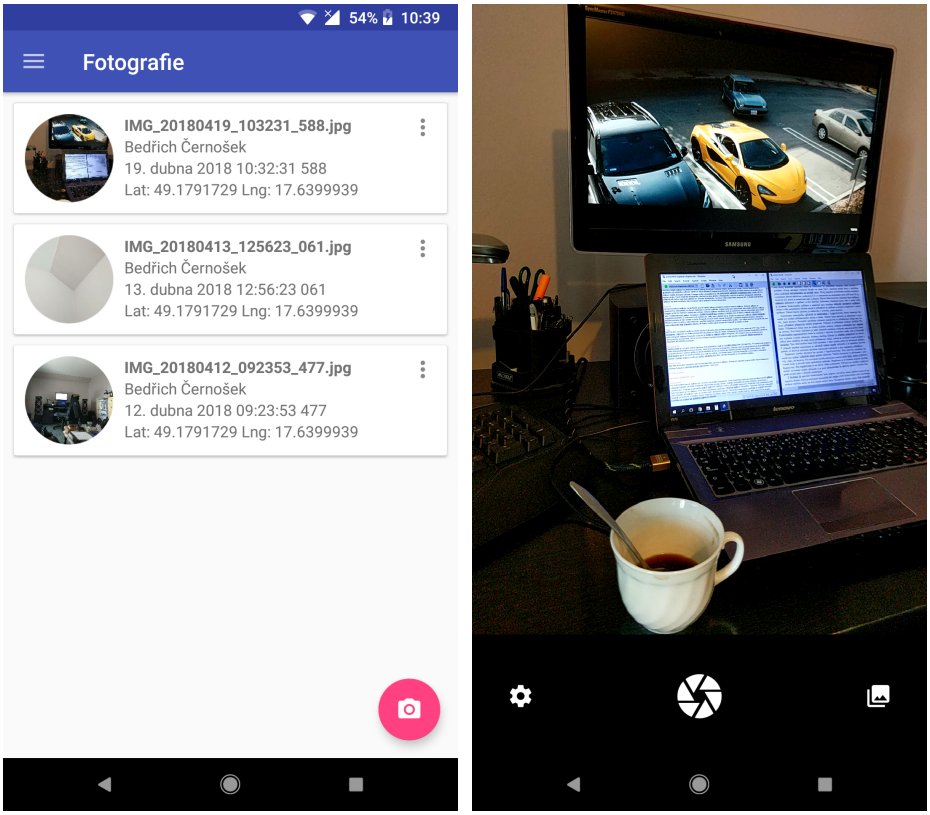
GalleryActivity, ve které je zobrazen snackbar informující uživatele o jeho úspěšném přihlášení.

Registrace Registraci nového uživatele lze provést v SignUpActivity. Tato aktivita obsahuje formulář pro zadání všech údajů nového uživatele. Vzhled obrazovky je předepsán v activity_sign_up layoutu. Pro vytvoření nového uživatele je nutné zadat login, jméno, příjmení, e-mail, heslo a potvrdit heslo opětovným zadáním. Registrace je provedena stiskem tlačítka Registrovat. Pro registrování je volán požadavek na server. Uživatel po úspěšné registraci obdrží objekt počátečního nastavení a je přesměrován do GalleryActivity modulu Fotografie stejně jako v případě autentizace.

Menu Menu aplikace bylo vytvořeno pomocí navigation drawer, které je dostupné vysunutím tzv. swipe z levé boční strany nebo otevřením stiskem tlačítka menu na levé straně action baru. Navigation drawer menu bylo optimalizováno pro použití s fragmenty a aktivitami. Byla proto vytvořena třída DrawerActivity pro navigation drawer, kterou v případě potřeby zobrazení menu v dané aktivitě tato aktivita jednoduše rozšiřuje. Klientská aplikace je rozdělena do pěti základních modulů, které jsou dostupné přes navigation drawer menu. Navigation drawer se skládá ze dvou částí. První je hlavička tzv. header, který obsahuje informace o aktuálně přihlášeném uživateli. Hlavička obsahuje miniaturu profilové fotografie, jméno a e-mail uživatele. Vzhled hlavičky je samostatně definován drawer_header layoutem. Zbytek draweru tvoří menu, které je definováno v menu_drawer. Celkový vzhled navigation draweru je pak definován v activity_drawer layoutu. Pro správné a bezproblémové zobrazení action baru bylo vhodné použít CoordinatorLayout a definovat vlastní Toolbar. Klíčové je vhodné nastavení šablony tzv. theme a nastavení chování layoutu obsahu, do kterého je pak vkládán obsah rozšiřující aktivity. Menu se skládá z modulů Fotografie, Uživatel, Nastavení, Klíče a Odhlášení. Ikony jednotlivých prvků menu jsou použity z material designu společnosti Google. Miniaturu uživatelské profilové fotografie uživatel vytváří v modulu Uživatel úpravou uživatelských informací. Kulatá miniatura profilové fotografie je vykreslena pomocí Glide.

Fotografie Modul Fotografie je tvořen aktivitou GalleryActivity, která obsahuje seznam všech vytvořených fotografií. Hlavní akcí tohoto modulu je pak vytvoření nové fotografie. Tato funkcionalita je dostupná z FloatingActionButton umístěného v pravém dolním rohu. Seznam fotografií je umístěn v GalleryFragment a je implementován pomocí RecyclerView pro načítání dat pouze aktuálně zobrazných prvků. Vzhled obrazovky je definován fragment_gallery layoutem. Pro naplnění seznamu daty je definován adapter seznamu PhotoListAdapter. Adapter je zodpovědný za provázání prvků seznamu s layoutem a předání hodnot. Zároveň taky spravuje hodnoty v seznamu. Při spuštění fragmentu dochází k volání požadavku na server, který vrací seznam fotografií. Seznam je poté zobrazen pomocí adapteru. Položka seznamu reprezentuje jednu fotografii vytvořenou pomocí aplikace. Vzhled jednotlivých položek v seznamu je definován list_photo_item layoutem prvku fotografie. Položka prvku seznamu tzv. item obsahuje miniaturu fotografie, název souboru, jméno a příjmení uživatele, časové razítko ze zařízení a získané GPS souřadnice. Součástí každé položky seznamu je ještě PopupMenu. Toto menu obsahuje možnosti zobrazení fotografie, zobrazení podrobností fotografie, verifikaci kryptografických prvků fotografie, rozpoznání registračních značek na fotografii a smazání fotografie. Fotografii lze zobrazit také výběrem požadované položky v seznamu bez nutnosti otevírat PopupMenu. Zobrazení miniatury

fotografie je implementováno pomocí Glide. Ten umožňuje také pomocí transformace vytvářet kulaté miniatury fotografie. Nad samotným seznamem je ještě implementován SwipeRefreshLayout, který umožňuje tažením z vrchu dolů tzv. swipe provést aktualizaci seznamu. Dojde k opětovnému volání požadavku na server a aplikace obdrží aktuální seznam fotografií.



(a) Galerie (b) Fotoaparát

Obrázek 4.3: Galerie a fotoaparát

Fotoaparát Fotografování je implementováno pomocí Android Camera2 API v CameraActivity resp. v jejím fragmentu CameraFragment. Vzhled fotoaparátu je definován pomocí fragment_camera layoutu a je také zvláště definovaný tzv. land layout fragmentu kamery pro fotografování na šířku tzv. landscape režim. Výchozí animace nastavená mezi jednotlivými layouty při otočení obrazovky byla vypnuta. Fotoaparát obsahuje pouze tři základní tlačítka. Tlačítko pro fotografování, nastavení fotoaparátu a galerii. Tato tlačítka se nachází ve spodní části obrazovky nad černým pozadím. Zbytek obrazovky je již vyhrazen pro TextureView a náhled tzv. preview fotoaparátu. K těmto účelům byl ukryt horní status bar a action bar. Nastavení fotoaparátu se provádí v aktivitě CameraSettingsActivity. Vzhled aktivity je definován v activity_camera_settings layoutu. V nastavení fotoaparátu je dostupné nastavení rozlišení fotografie. Rozlišení fotografie je plně ponecháno na uživateli. Server toto nastavení nijak neovlivňuje a nezasahuje do něj. Velikost rozlišení však ovlivňuje rychlost zpracování fotografie. Uložení nastavení se provádí pomocí tlačítka Uložit. Tlačítko

galerie ve fotoaparátu slouží pro návrat na seznam fotografií. Tlačítko pro fotografování zachytí fotografii a spustí proces zpracování obrazu a získání kryptografických prvků fotografie.

Fotografování Pro správu a přístup k fotoaparátu zařízení se využívá CameraManager. Nastavení možností a vlastností fotoaparátu se provádí pomocí CameraCharacteristics. Kromě zobrazení náhledu pomocí TextureView a SurfaceTexture je nutné také zajistit cílový prostor neboli plochu pro zachycení fotografie. K tomuto účelu slouží ImageReader. Při zachycení fotografie se vytváří CaptureRequest, který definuje veškeré parametry pro vytvářený snímek. Jakmile je vytvořen požadavek, tak může být vytvořena CaptureSession. Po zpracování požadavku kamera vrátí TotalCaptureResult objekt, který obsahuje informace o stavu kamery v průběhu vytvoření snímku a použité nastavení. Jakmile je snímek pomocí ImageReader dostupný, tak pomocí listeneru může být fotografie z objektu třídy Image zpracována.

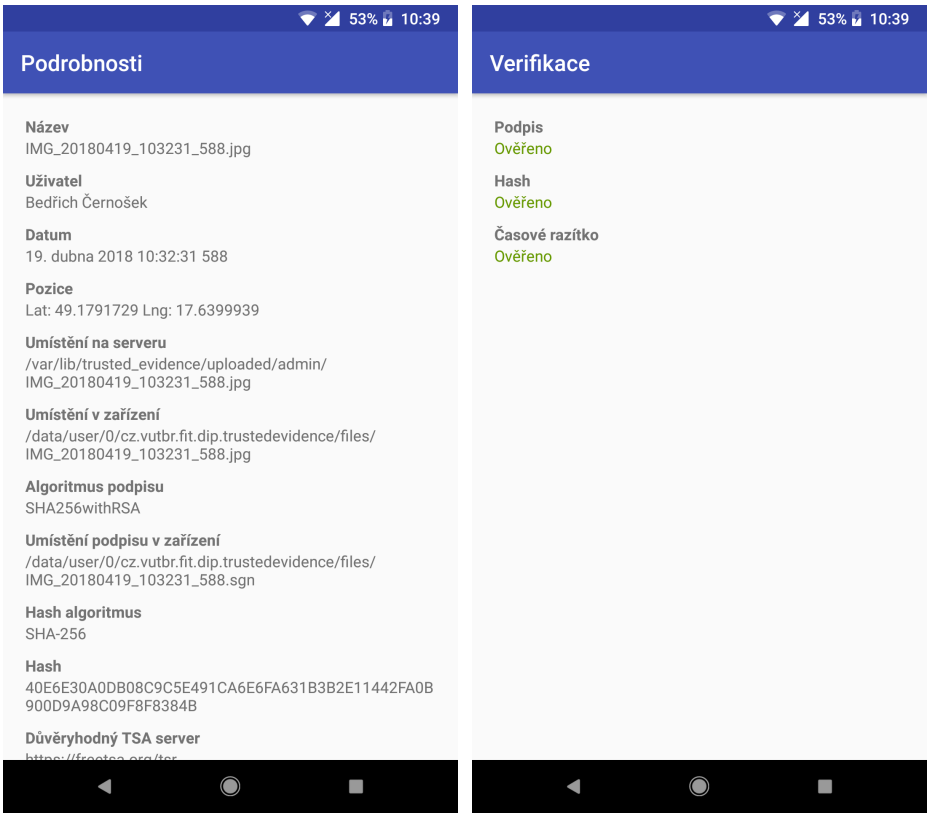
Kryptografické prvky Po vytvoření fotografie jsou získávány jednotlivé kryptografické prvky fotografie. V zařízení uživatele je vytvořeno časové razítko. Fotografie je poté podepsána algoritmem SHA256withRSA pomocí soukroméno RSA klíče vygenerovaného uživatelem. Z původní fotografie je vytvořen SHA-256 hash a jsou získány GPS souřadnice zařízení. Pro zobrazení seznamu fotografií je ještě vytvořena miniatura fotografie. Miniatura fotografie je velikosti 200x200 pixelů. Při vytváření miniatury byla využita komprese a kvalita výsledné miniatury byla snížena na 75%. Byl také použit datově méně náročný model ARGB8888. Fotografie a její podpis je uložen do external storage v paměti mobilního zařízení. Následně je vytvořen objekt fotografie, do kterého se vloží uživatel, časové razítko ze zařízení, hash, podpis, GPS souřadnice, miniatura fotografie a odešle se spolu s původní fotografií společně jedním voláním požadavku na server.

Podrobnosti fotografie Podrobnosti fotografie jsou zobrazeny ve PhotoDetailActivity. Ta pouze přijme objekt fotografie od fragmentu galerie a přehledně zobrazí data. Vzhled je definován pomocí activity_photo_detail layoutu. Zobrazeny jsou zde veškerá dostupná data fotografie. Uživatel má tedy přehledně k dispozici název souboru, jméno a příjmení uživatele, časové razítko ze zařízení, GPS souřadnice, umístění fotografie na serveru, umístění fotografie v zařízení, algoritmus podpisu, umístění podpisu v zařízení, hash algoritmus, hash, použitý důvěryhodný TSA server a časové razítko od důvěryhodného serveru.

Verifikace Verifikace kryptografických prvků fotografie se provádí z PhotoVerificationActivity. Její vzhled je definovaný v activity_photo_verification layoutu. Tato aktivita volá požadavek na server a zobrazuje výsledek, který vrací server. Výsledek verifikace se nachází v objektu Verification, který obsahuje výsledky verifikace všech kryptografických prvků fotografie. Verifikace probíhá na serveru a bude popsána v příslušné podkapitole.

Detekce a rozpoznání objektů v obraze Detekce a rozpoznání registračních značek zachycených na pořízené fotografii je prováděno pomocí JavaANPR. Rozpoznání je dostupné z PopupMenu prvku seznamu fotografií. Detekce a rozpoznání je prováděno z PhotoRecognitionActivity. Vzhled aktivity se nachází v activity_photo_recognition layoutu. Je volán požadavek na server, který provádí detekci a rozpoznání objektů na

fotografii umístěné na serveru a vrací řetězec s rozpoznanou registrační značkou. JavaANPR používá pro detekci registrační značky detekci hran a konvoluční matice. Před rozpoznáním provádí segmentaci, extrakci vlastností a normalizaci registrační značky. Pro rozpoznání registračních značek používá neuronové sítě.



(a) Podrobnosti fotografie (b) Verifikace kryptografických prvků

Obrázek 4.4: Podrobnosti a verifikace kryptografických prvků fotografie

Uživatel Pro správu uživatelského účtu slouží modul Uživatel, který je dostupný z navigation drawer menu. V UserActivity resp. UserFragment je uživateli zobrazen uživatelský profil, ve kterém nelezne veškeré údaje. Vzhled je definován fragment_user layoutem. Graficky se tato obrazovka skládá ze dvou částí. Hlavička obrazovky zobrazuje miniaturu uživatelské profilové fotografie nad barevným pozadím. Spodní část zobrazuje jednotlivé údaje uživatele. Zobrazeno je jméno a příjmení uživatele, login, e-mail a role uživatele v systému. Hlavní akcí této obrazovky je úprava uživatelského profilu, která je dostupná přes FloatingActionButton.

Úprava uživatele Upravit uživatele je možné pomocí UserEditActivity. Tato obrazovka je vzhledově velmi podobná předchozí obrazovce pro zobrazení uživatelského profilu. Obsahuje však editovatelné prvky EditText a možnosti pro výběr, vytvoření a mazání profilové fotografie. Uživatelskou fotografii lze vybrat z galerie fotografií v mobilním zařízení. Také je možné vytvořit fotografii novou pomocí fotoaparátu zařízení. Výběr fotografie a fotografování je realizováno standardně pomocí intentů. U uživatelské fotografie není kladen nárok na bezpečnost a kryptografické prvky. Je tedy vhodné

použít tento mnohem snadnější a rychlejší způsob výběru či pořízení fotografie. V této aktivitě lze upravit jen podmnožina údajů uživatele. Uživatel v rámci mobilní aplikace nemá právo upravit roli v systému nebo přihlašovací login. Upravit lze tedy pouze profilovou miniaturu fotografie, jméno, příjmení a e-mail uživatele. Uložení upravených údajů se provádí stiskem tlačítka Uložit. Uložení je realizováno voláním požadavku na server s objektem uživatele. Server již zajistí aktualizaci objektu v rámci databáze systému. Vzhled obrazovky je definován activity_user_edit layoutem.

Nastavení od serveru Pro zobrazení nastavení aplikace slouží modul Nastavení. Tento modul je také dostupný z navigation drawer menu. V rámci tohoto modulu je spuštěna SettingsActivity. Obsah se nachází v SettingsFragment a vzhled obrazovky je definován ve fragment_settings layoutu. Na této obrazovce lze nalézt nastavení definované pomocí mechanismu nastavení při počáteční komunikaci. Počáteční nastavení je přijato od serveru při registraci a přihlášení. Je reprezentováno objektem třídy InitialSetup, která bude popsána v podkapitole Datový model. Klient přijaté nastavení uloží a používá jej v rámci mobilní aplikace. Jedná se o nastavení od serveru, které nelze v aplikaci měnit. Toto nastavení definuje úložiště fotografií, algoritmus podpisu, hash algoritmus a důvěryhodný server poskytující časové razítko.

Klíče Správa RSA klíčů pro digitální podpis je prováděna v modulu Klíče. Modul je také dostupný z navigation drawer menu a spouští KeysActivity. Obsah je umístěn v KeysFragment a vzhled ve fragment_keys layoutu. Obrazovka umožňuje nahrát veřejný klíč na server pro verifikaci podpisu z webové aplikace. Dále zajišťuje export RSA klíčů do external storage a import RSA klíčů z external storage.

Odhlášení Poslední prvek navigation drawer menu je možnost odhlášení. Ta již nespouští žádné další aktivity ani fragmenty a netvoří ji žádné layouty. Tato funkcionalita pouze v listeneru provádí nezbytné úkony před ukončením uživatelské činnosti před samotným odhlášením. Po odhlášení jsou ukončeny veškeré aktivity a fragmenty v hierarchii spuštěných obrazovek a uživatel je následně přesměrován na hlavní main aktivitu.

REST API klient Komunikace s RESTful API serverové webové aplikace je na straně klienta zajištěna pomocí Retrofit2 klienta. Mezi serverovou a klientskou stranou se přenášejí data ve formátu JSON. Pro konverzi dat mezi JSON a objekty tzv. POJO je vhodné použít například Gson converter. Retrofit2 používá pro HTTPS požadavky klienta OkHttpClient. V Android projektu je nutné přidat do build.gradle implementation pro Retrofit2, OkHttpClient a Gson converter v dependencies. Retrofit2 klient je implementován ve třídě RetrofitClient, který je umístěn v balíčku rest vytvořeném pro klienta, providery a model. Retrofit2 klient je vytvářen voláním statické metody createService, která volá funkci getRetrofit, která je zodpovědná za sestavení a vrácení objektu Retrofit. Sestavení probíhá pomocí metody buildRetrofit, kde je také nastavena URL adresa serveru API, sestaven HTTP klient a přidána Gson converter factory. Sestavení HTTP klienta pomocí metody buildClient umožňuje také nastavit timeout spojení, timeout zápisu, timeout čtení a LoggingInterceptor. Vytvoření converter factory v metodě createConverterFactory zahrnuje nastavení fotmátu času a registraci serializérů a deserializérů. V rámci tohoto projektu byly vytvořeny vlastní deserializery pro pole bytů, Date a Timestamp. Každá entita ukládaná v databázi má svůj vlastní provider, který implementuje rozhraní služby. Rozhraní služby pak obsahuje jednotlivá volání požadavků na API serveru.

4.3 Popis práce na serverové aplikaci

Serverová aplikace je vyvíjena v prostředí NetBeans 8.2. Pro vývoj serverové aplikace byl vytvořen nový projekt webové aplikace na platformě Java EE. Ve vytvořeném projektu byl konfigurován Maven. Webová aplikace se skládá z datového modelu aplikace, persistentní jednotky, služeb pro databázové operace, beanů, jersey servletu, služeb API, filtrů, converterů, služby pro zasílání e-mailů, služby pro získání časového razítka od důvěryhodného serveru a služby pro verifikaci kryptografických prvků. Beany webové aplikace provádějí požadované výpočty a operace, které jsou snadno pomocí těchto beanů zobrazovány na webových stránkách pomocí technologie Java Server Pages zkráceně JSP. Webové stránky jsou ve formátu XHTML a zodpovědná za zobrazení webové prezentace je taktéž tato aplikace. Front-end webové aplikace je tvořen pomocí technologie Java Server Faces zkráceně JSF a frameworků. Využit je v rámci webové aplikace framework Bootsfaces 1.2.0 a Primefaces 6.0. Aplikace komunikuje pomocí služeb s Jersey servletem, na kterém běží RESTful API. Toto API je implementováno pomocí technologií JAX-RS a Jersey 2. Jednotlivé požadavky API jsou detailně popsány v podkapitole Komunikace. V rámci serveru běží také databázový systém MySQL. Webová aplikace s databázovým systémem komunikuje pomocí technologie JDBC.

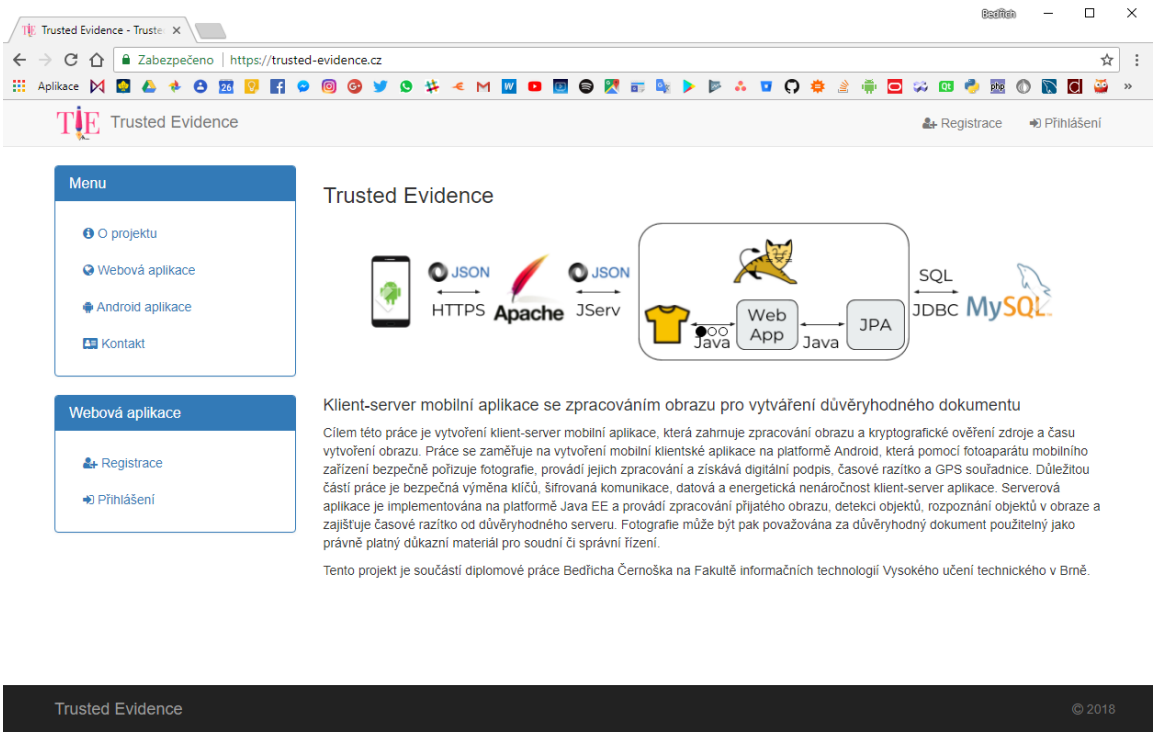
Databáze Konkrétně se o spojení stará Hibernate přes connection pool HikariCP. Pro mapování objektů v MySQL je využíváno Java Persistence API zkráceně JPA. Spojení s databází je definováno persistentní jednotkou v souboru persistence. Jsou zde nastaveny přístupové údaje k databázi, parametry spojení, použitý driver a poskytovatel spojení. Další potřebné třídy se nacházejí v balíčku persistence. Komunikace s databází je zajištěna vytvořením objektu EntityManager pomocí factory. Každá entita pak využívá EntityManager pro provádění potřebných databázových operací. Služba tzv. service každé entity se skládá ze tří základních částí. První je rozhraní tzv. interface služby, který rozšiřuje DomainService. Tato služba poskytuje rozhraní k běžně používaným databázovým operacím. Rozhraní služby může definovat další vlastní databázové operace. Druhou částí služby je obecná služba, která rozšiřuje JPAService pro danou entitu a implementuje rozhraní. V obecné službě jsou umístěny vlastní dotazy tzv. query do databázového systému. Třetí částí je implementace služby, která rozšiřuje obecnou službu a je zodpovědná za získání objektu EntityManager. Jednotlivé služby tzv. services jsou pak spravovány pomocí třídy MySQLServiceManager.

Datový model a beany Byl implementován datový model a byly vytvořeny beany pro každou entitu datového modelu. Tvorba datového modelu je popsána v podkapitole Datový model. Pro autentizaci byl vytvořen LoginBean. Bean při přihlašování ověřuje login a heslo uživatele. V případě úspěšného přihlášení uživatele přesměruje na definovanou adresu. Přihlášení je platné v rámci session. V případě vypršení session je uživatel odhlášen. Manuální odhlášení zneplatní běžící session. LoginBean je využíván pouze v rámci webové aplikace. Autentizace u klientské mobilní aplikace je vyřešena přes API. Pro registraci ve webové aplikaci slouží SignUpBean, který vytváří nového uživatele.

Front-end Webové stránky ve webové aplikaci používají template, který importuje veškeré potřebné styly, skripty a knihovny. Template je strukturován jako běžný XHTML dokument. Webové stránky jsou pak pouze složeny z titulku a obsahu, který je vkládán do template. Webové stránky používají prvky JSF knihovny Bootsfaces a Primefaces.

Tělo XHTML stránky je tvořeno horní navigační lištou tzv. navBar, který obsahuje logo a název aplikace v levé části. Pravá část pak obsahuje informaci o přihlášeném uživateli a možnost se přihlásit či odhlásit. Odkazy jsou implementovány pomocí navLink či navCommandLink. Dále je tělo tvořeno kontejnerem, který v levé části obsahuje panely menu a v pravé části je poté prostor pro obsah, který je vkládán z webové stránky. Spodní část těla je tvořena také pomocí navBar, který obsahuje název projektu, copyright a rok vytvoření aplikace. Webové stránky a framework Bootsfaces je založen na Bootstrapu a tudíž jsou stránky responzivní a zobrazitelné v každém zařízení. Webové stránky jsou rozděleny na část veřejnou, která je umístěna v adresáři public. Druhá část jsou privátní stránky, které jsou umístěny v adresáři pages. Do veřejné části patří hlavní stránka O projektu, stránka klientské aplikace Android aplikace, stránka serverové aplikace Webová aplikace a stránka Kontakt. Součástí veřejných stránek je i template, stránka Přihlášení a stránka Registrace. Privátní část obsahuje Dashboard se statistikami, správu fotografií a správu uživatelů.

Hlavní stránka Hlavní stránka main se skládá ze dvou částí. Ve vrchní části stránky se nachází prezentace obrázků klientské a serverové aplikace. Prezentace je implementována pomocí carousel z Bootsfaces frameworku. Oproti standardní verzi z frameworku je použitý carousel částečně upravený. Je vytvořený vlastní styl pro indikátory a ovládací prvky prezentace. Spodní část obrazovky pak tvoří panelGrid s odstavci textu. První odstavec textu obsahuje abstrakt práce. Druhý odstavec textu obsahuje informace o projektu.



Obrázek 4.5: Hlavní stránka

Stránka webové aplikace Stránka Webová aplikace je tvořena pomocí panelGrid z Bootsfaces frameworku. Tento panelGrid obsahuje dva sloupce s informacemi o webové

aplikaci. V prvním sloupci se nachází v bodech funkcionalita webové aplikace. Druhý sloupec obsahuje v bodech parametry webové aplikace.

Stránka klientské aplikace Informace ohledně klientské aplikace jsou na stránce Android aplikace. Tato stránka je koncipována obdobně jako stránka o webové aplikaci. Je opět tvořena pomocí panelGrid, který se skládá ze dvou sloupců. První sloupec obsahuje v bodech informace o funkcionalitě klientské Android aplikace. Ve druhém sloupci jsou parametry mobilní aplikace.

Stránka kontakt Stránka Kontakt obsahuje kontaktní informace na autora práce. Mezi kontaktními informacemi lze nalézt jméno, příjmení, telefon a e-mail na autora. Tato stránka je tvořena pomocí panelGrid. Skládá se z odstavce obsahující uvedené informace.

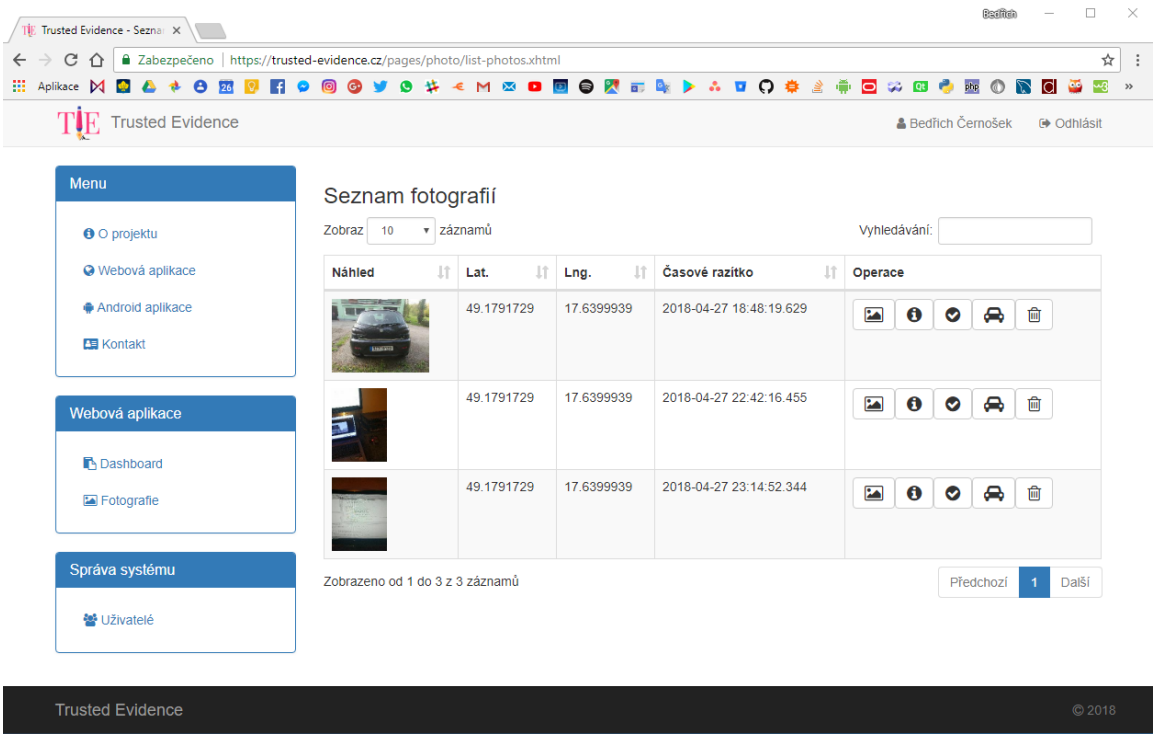
Stránka přihlášení Autentizační stránka obsahuje formulář pro přihlášení. Tato stránka nepoužívá zmíněný template. Přihlášení je provedeno pomocí LoginBean. Formulář je složen z container, panel, inputText, inputSecret a commandButton prvků frameworku Bootsfaces.

Stránka registrace Registrace je koncipována obdobně jako přihlášení. Stránka Registrace obsahuje formulář pro vyplnění uživatelských údajů. Vyplňuje se jméno, příjmení, login, e-mail a heslo. Stránka také nevyužívá template. Registrace je provedena pomocí SignUpBean po kliknutí na tlačítko Registrovat. Stránka registrace používá shodné prvky jako stránka přihlášení.

Stránka dashboard Stránka Dashboard zobrazuje přehledně v dlaždicích statistiky systému a základní informace. Informace a upozornění jsou vytvořeny pomocí prvku alert z frameworku Bootsfaces. Stránka je dále členěna pomocí panelGrid. Dlaždice je implementována pomocí carousel z Bootsfaces. Carousel dlaždice má přepracovaný styl. Dlažice mění jednotlivé statistické údaje a ikonu modulu.

Fotografie Správa fotografií resp. modul Fotografie zahrnuje seznam fotografií, podrobnosti fotografie, verifikaci kryptografických prvků fotografie a mazání fotografie. Jakákoliv úprava fotografií není vůbec uvažována. Seznam fotografií je tvořen prvkem dataTable frameworku Bootsfaces. Každá položka seznamu obsahuje náhled fotografie, GPS souřadnice, časové razítko z mobilního zařízení a operace. Náhled fotografie je zobrazen pomocí graphicImage z frameworku Primefaces a využívá thumbnailBean, kterému předá id fotografie. Bean převede miniaturu z pole bytů do StreamedContent, který lze pomocí graphicImage zobrazit. Informace položky seznamu jsou získány z PhotoBean a zobrazeny do dataTableColumn. Tlačítka operací jsou implementována pomocí commandButton. Operace jsou vykonávány pomocí PhotoBean.

Zobrazení fotografie Mezi operace, které lze provádět s položkou seznamu, patří zobrazení fotografie. Stránka je rozčleněna pomocí panelGrid. Zobrazení fotografie je realizováno pomocí graphicImage. Využívá se zde LocalImageBean, který načte fotografii uloženou v souborovém systému serveru. Fotografii bean převede do StreamedContent, který je zobrazitelný v graphicImage. Využita je maximálně šířka dostupná pro zobrazení obsahu.



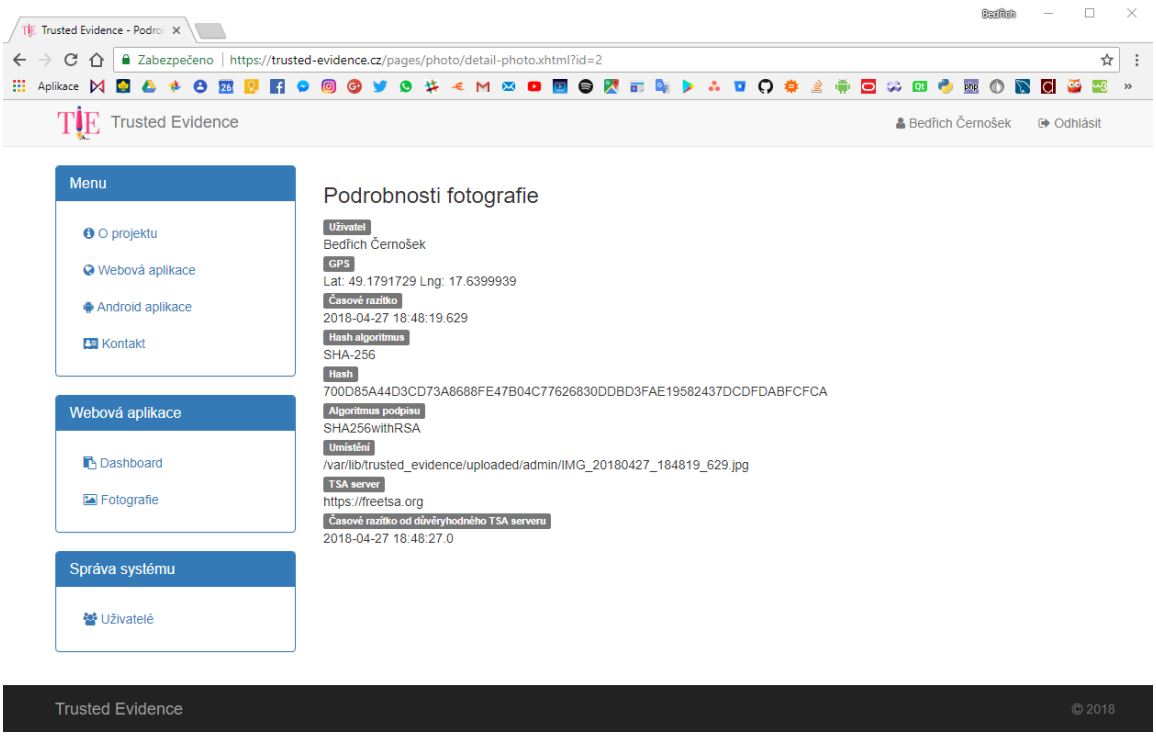
Obrázek 4.6: Seznam fotografií

Podrobnosti fotografie Další operací je zobrazení podrobností fotografie. Stránka používá panelGrid a jednotlivé řádky jsou umístěny ve sloupci využívající šířku dostupnou pro zobrazení obsahu. Zobrazeny jsou informace o uživateli, GPS pozici, časovém razítku mobilního zařízení, hashi, podpisu a časovém razítku od důvěryhodného serveru. Nadpis je implementován pomocí label a hodnota pomocí outputText.

Verifikace V dostupných operacích lze nalézt také možnost verifikace kryptografických prvků fotografie. Na začátku je spuštěna metoda verify z PhotoBean, která zajišťuje ověření. Výsledek je zobrazen ve sloupci v panelGrid. Metoda verify pak využívá VerificationService, která provádí ověření podpisu. Podpis se ověřuje porovnáním veřejného klíče uživatele, podpisu a původní fotografie. Třída VerificationService provádí také ověření hashe. Vytváří hash z původní fotografie a porovnává jej s existujícím hashem. Ověření časového razítka od důvěryhodného serveru provádí TimestampService porovnáním hashe a tokenu.

Detekce a rozpoznání objektů v obraze Z operací seznamu fotografií lze také provádět rozpoznání registrační značky vozidla na fotografii. Nejprve je provedena metoda recognize, která získá z objektu fotografie cestu k souboru. Metoda načte fotografii ze souborového systému serveru. Poté provádí detekci a rozpoznání registrační značky na načtené fotografii pomocí JavaANPR. Výsledek je zobrazen v panelGrid.

Mazání Poslední operací dostupnou z položky seznamu fotografií je operace mazání fotografie. Mazání provádí photoBean smazáním fotografie pomocí databázové služby a načtením nového seznamu fotografií.



Obrázek 4.7: Podrobnosti fotografie

Uživatelé Správa uživatelských účtů je dostupná pouze pro uživatele s rolí administrátora. Implementovaná správa poskytuje seznam uživatelů, vytvoření nového uživatele, úpravu uživatele a mazání uživatele. Seznam uživatelů je umístěn v dataTable, kde hodnoty jsou vloženy do jednotlivých sloupců dataTableColumn. Tlačítka operací jsou vytvořena pomocí commandButton. Všechny operace související se správou jsou implementovány v UserBean. V případě otevření stránky pro úpravu uživatele nebo vytvoření nového uživatele se požadované přesměrování tzv. redirect provádí pomocí beanu. Databázové operace ukládání a mazání taktéž provádí bean pomocí databázových služeb.

Úprava uživatele Přihlášený uživatel má možnost upravit svůj uživatelský účet. Upravit však může pouze jméno, příjmení, heslo a e-mail. Parametry uživatele jako je například login a role není povoleno uživateli s uživatelskou rolí upravovat. Formulář je rozčleněn pomocí panelGrid a složen z inputText a inputSecret prvků. Tlačítko uložení tvoří commandButton. Uložení provádí bean pomocí databázové služby.

REST API server Komunikace mezi klientskou mobilní aplikací a serverovou webovou aplikací je vyřešena pomocí RESTful API. Webová aplikace používá technologie JAX-RS a Jersey 2, jejichž dependency musí být vložena v xml souboru pom. V xml souboru web je pak nutné definovat jersey servlet. Servlet je konfigurován pomocí parametrů. K běhu servletu je potřeba nastavit balíček, kde jsou umístěny jednotlivé služby v rámci webové aplikace. Pokud má RESTful API využívat filtry, je nutné každý z těchto filtrů uvést v konfiguraci servletu. Tímto způsobem jsou konfigurovány filtry v API pro autentizaci a kontrolu rolí. Filtry jsou detailně popsány dále. Vytvořené fotografie jsou přenášeny přes RESTful API jako multipart. Multipart funkcionalitu

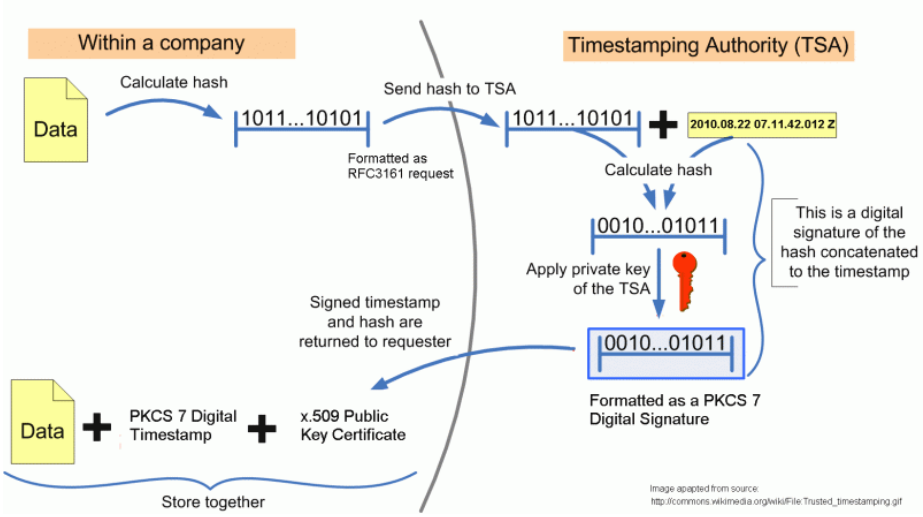
je nutné také definovat jako parametr v konfiguraci API. Pro API je vytvořen balíček rest, kde jsou umístěny všechny služby. RestService je služba API běžící na výchozí URL adrese API. Tato služba pouze informuje uživatele o provozu API a vrací čas serveru. Ostatní služby jsou členěny dle entit v systému. Služba pro uživatele se nazývá UserRestService a nabízí požadavky tzv. requests pro operace související s uživatelem. PhotoRestService je služba pro fotografie. Zajišťuje veškeré operace týkající se fotografie včetně jejich odesílání na server. Jednotlivé požadavky budou podrobněji popsány v podkapitole Komunikace.

Filtry Každý filtr implementuje ContainerRequestFilter a přepisuje metodu filter, kde je filtrování implementováno. Důležité je přidat anotaci Provider pro třídu filtru. Anotace Provider definuje rozšíření a přizpůsobení JAX-RS běhu. Vhodné je také definovat pre matching pomocí anotace PreMatching v případě provedení filtru před zpracováním požadavku. Post matching je výchozí a není nutné jej explicitně nastavovat. Filtr může mít nastavenou i různou prioritu pro upřednostnění mezi ostatními filtry. Důležitým filtrem ve webové aplikaci je AuthRestFilter, který je zodpovědný za autentizaci uživatele v rámci RESTful API a tedy v rámci komunikace klientské a serverové aplikace. Autentizační filtr se provádí před zpracováním požadavku a má nastavenou prioritu pro autentizaci. Filtr zjišťuje pomocí ContainerRequestContext metodu a URL požadavku. Také získává z hlavičky HTTPS požadavku řetězec Authorization s přihlašovacími údaji. Na základě získaných dat z kontextu je poté filtr schopen udělovat výjimky pro vybrané metody a URL. Na požadavky, kde není vyžadována autentizace není uplatňován tento filtr. V případě nutnosti autentizace filtr vyvolá příslušnou výjimku, pokud je autorizační řetězec prázdný. Filtr provádí parsování autorizačního řetězce. Získává z něj login a heslo, které následně ověřuje v databázi. Pokud přihlašovací údaje nesouhlasí je vyvolána výjimka. Filtr ještě ukládá uživatele získaného z databáze do HttpServletRequest. Takto získaný uživatel se využije při zpracování požadavku, aby se nemuselo provádět jeho opětovné získání z databáze. Filtr pro ověření uživatelské role AllowedRolesFilter je prováděn po zpracování požadavku. Využívá uloženého uživatele z HttpServletRequest. Pomocí ResourceInfo je filtr v přepisované metodě filter schopen získat seznam povolených rolí v požadavku pomocí anotace AllowedRoles. Filtr prochází seznam a zjišťuje, zda se role uživatele nachází v tomto seznamu. Pokud se role uživatele v tomto seznamu nenachází, pak filtr vyvolá výjimku.

Zasílání e-mailů Zasílání e-mailů z webové aplikace je prováděno pomocí JavaMail API. Odeslání e-mailu se v aplikaci využívá pro zaslání zapomenutého hesla uživateli. Pro použití zmíněného API je nutné přidat dependency do xml souboru pom. Pro zasílání e-mailu byla vytvořena třída SendMail. Nejprve je nutné v získaném objektu třídy Properties nastavit SMTP parametry. Pro správné fungování je nutné povolit SSL a autentizaci. Dále je potřeba nastavit host, adresu odesílatele, adresu příjemce a port. Na serveru je taky klíčové zkontrolovat a případně povolit přenos dat na tomto portu ve firewall. Po nastavení properties je vytvořena Session a MimeMessage. Do MimeMessage je nutné nastavit odesílatele, příjemce, předmět zprávy a obsah zprávy. Odeslání je prováděno pomocí třídy Transport.

Časové razítko od důvěryhodného serveru Získání časového razítka od důvěryhodného serveru je implementováno ve třídě TimestampService. Pro komunikaci s důvěryhodným TSA serverem je využito Bouncy Castle API. Pro použití API je nutné

přidat dependency do xml souboru pom. Časové razítko od důvěryhodného serveru se získává ihned po získání fotografie a objektu fotografie od klientské mobilní aplikace. Proces získání časového razítka od důvěryhodného serveru začíná vytvořením identifikátoru použitého algoritmu. Poté se vytváří MessageImprint, pomocí hashe fotografie a identifikátoru algoritmu. Následně je již vytvářen požadavek na TSA server. Ve webové aplikaci je využíván server freetsa.org a není explicitně nastavena žádná politika serveru. Parametry vytvářeného požadavku jsou MessageImprint, OID politiky serveru, bezpečně vygenerované náhodné číslo tzv. nonce, boolean hodnota pro použití certifikátu a použítá rozšíření. Jakmile je vytvořen požadavek vytváří se UrlConnection, kde je nastaven timeout, provádění vstupu a výstupu, cachování, content type a kódování. Požadavek je zapsán do output streamu spojení. Pomocí input streamu spojení je získaná odpověď tzv. response v poli bytů. Pole je poté parsováno a převedeno na TimeStampResponse. Následně se kontrolují chyby a provádí se ověření správnosti odpovědi. Z objektu TimestampResponse lze již získat chybové kódy, token a timestamp. Token a timestamp je vložen do objektu vytvořené třídy pro návrat výsledku. Token a timestamp je vhodné uložit do databáze pro verifikaci časového razítka. Ověření časového razítka může být poté kdykoliv provedeno. Ověření se provádí porovnáním hashe fotografie a tokenu. Z tokenu se získají podepsaná data, která jsou převedena na TimestampToken. Poté je získán certifikát TSA serveru, jehož platnost je ověřena. Nakonec je z TimestampTokenInfo získán hash, který je porovnáván s hashem původní fotografie. Pokud hashe souhlasí, časové razítko je platné. V případě neúspěchu při porovnání je vyvolána výjimka.

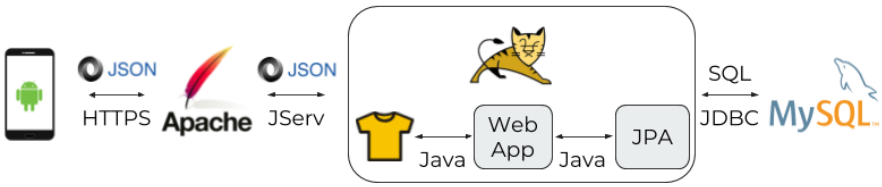


Obrázek 4.8: Komunikace s TSA serverem[46]

4.4 Komunikace

Komunikace mezi klientskou mobilní aplikací a serverou webovou aplikací je realizována pomocí RESTful API, které je implementováno ve webové aplikaci na straně serveru. Požadavky tzv. requests tohoto API jsou vykonávány pomocí služeb tzv. services. Každá entita datového modelu má vytvořenou vlastní službu v balíčku rest. Služby provádějí přímo databázové operace pomocí databázových služeb. Metoda HTTPS požadavku je definována

pomocí identicky pojmenované anotace. Vstup a výstup resp. konzumovaný objekt a produkovaný objekt požadavku je definovaný anotacemi Produces a Consumes. Konkrétní URL cesta je definována pomocí anotace Path. Povolené role jsou definované pomocí anotace RolesAllowed.



Obrázek 4.9: Komunikace

Výchozí služba Mimo REST služby entit datového modelu je vytvořena ještě služba RestService. Tato služba je provozována na výchozí adrese API a obsahuje jediný GET požadavek produkující HTML dokument s UTF-8 kódováním informující o fungování a provozu API. Služba také vrací čas serveru.

Požadavky týkající se entity uživatele datového modelu jsou implementovány v UserRestService. Služba využívá URL prefix **user**.

Přihlášení Pro přihlašování slouží požadavek `getUser`, který používá URL suffix **login**. Parametr požadavku je řetězec obsahující login uživatele. Z databáze je získán objekt uživatele s požadovaným loginem z parametru. Login je unikátní vlastnost entity uživatele. Identifikátor získaného objektu uživatele z databáze je ještě porovnán s identifikátorem objektu uživatele z autentizace. Požadavek produkuje objekt počátečního nastavení třídy InitialSetup.

Registrace Registrace se v API provádí požadavkem `signUpUser` s URL suffixem **signup**. Vstupním objektem je objekt uživatele, který je uložen do databáze. Kontrolována je role nového uživatele. Požadavek pak produkuje objekt počátečního nastavení třídy InitialSetup.

Zapomenuté heslo Zapomenuté heslo se získá pomocí požadavku `resetPassword` s URL suffixem **lost**. Požadavek konzumuje řetězec obsahující e-mail uživatele. E-mail je taktéž unikátní vlastnost v tabulce uživatele. Objekt uživatele je nalezen v databázi podle e-mailu z těla požadavku. Uživateli je odeslán e-mail s přihlašovacími údaji přes službu implementovanou pomocí JavaMail API.

Získání objektu uživatele Pro získání objektu uživatele se používá požadavek `getUser` s URL suffixem **get/me**. Parametrem požadavku je identifikátor objektu uživatele. Objekt uživatele je získán z databáze pomocí identifikátoru. Identifikátor objektu uživatele z databáze je pak porovnán s identifikátorem objektu uživatele z autentizace. Výstupem požadavku je získaný objekt uživatele.

Získání objektu uživatele administrátorem Existuje varianta požadavku pro získání libovolného objektu uživatele pro administrátora `getUserAdmin` s URL suffixem **get**. Na rozdíl od předchozího požadavku tento požadavek neprovádí kontrolu identifikátoru uživatele a navíc kontroluje roli uživatele pomocí filtru.

Seznam objektů uživatelů Získání seznamu objektů uživatelů je umožněno administrátorovi pomocí požadavku `getUsersList` s URL suffixem **list**. Požadavek získá seznam objektů z databáze. Seznam objektů uživatelů je výstupem tohoto požadavku. Role uživatele je kontrolována filtrem.

Uložení objektu uživatele Objekt uživatele je ukládán pomocí požadavku `postUser` s URL suffixem **post/me**. Vstupem je objekt uživatele. Identifikátor objektu uživatele je porovnán s identifikátorem objektu uživatele z autentizace. Poté je upravený objekt uživatele aktualizován v databázi. Výstupem je řetězec informující o výsledku operace.

Uložení objektu uživatele administrátorem Taktéž je vytvořena i varianta pro administrátora `postUserAdmin` s URL suffixem **post**. Pokud objekt uživatele již existuje, uživatelská data jsou upravena. Jinak je v databázi vytvořen nový objekt uživatele. Požadavek neprovádí žádné kontroly identifikátoru, pouze proběhne kontrola role uživatele pomocí filtru. Výstupem je také řetězec s výsledkem operace.

Uložení veřejného klíče Veřejný RSA klíč pro ověření digitálního podpisu na serveru je uložen pomocí požadavku `postPublicKey` s URL suffixem **key/my**. Vstupem je objekt třídy Keys, ve kterém se nachází pouze veřejný klíč. Veřejný klíč je uložen do objektu uživatele z autentizace. Objekt uživatele je poté aktualizován v databázi. Výstupem je objekt třídy Response.

S entitou fotografie datového modelu pracuje služba PhotoRestService, která používá URL prefix **photo**.

Nahrání fotografie Základním požadavkem je `uploadFileWithObject` s URL suffixem **upload/object**. Očekáván je na vstupu multipart obsahující původní fotografii a objekt fotografie jako JsonPart. Požadavek tedy konzumuje na vstupu multipart form data. Přijatý objekt ve formátu JSON je parsován a do vytvořeného objektu je vložena cesta k souboru fotografie na serveru. Kontrolován porovnáním je identifikátor objektu uživatele z objektu fotografie s identifikátorem objektu uživatele z autentizace. Fotografie je uložena do souborového systému serveru. Je kontrolována existence adresářů, které jsou případně vytvářeny. Následně je získáno časové razítko od důvěryhodného serveru. Časové razítko a token je vložen do objektu fotografie. Požadavek ukládá objekt do databáze a vrací řetězec s výsledkem operace.

Získání objektu fotografie Získat objekt fotografie uživatel může pomocí požadavku `getPhotoUser` s URL suffixem **get/my**. Parametrem požadavku je identifikátor objektu fotografie. Objekt fotografie je získán z databáze pomocí identifikátoru. Identifikátor objektu uživatele z objektu fotografie je porovnáván s identifikátorem objektu uživatele z autentizace. Požadavek vrací na výstup objekt fotografie.

Získání objektu fotografie administrátorem Pro administrátora je vytvořena verze požadavku na získání libovolného objektu fotografie. Rozdíl mezi požadavky je, že se nekontroluje identifikátor uživatele, ale kontroluje se role uživatele. Požadavek `getPhotoAdmin` se suffixem **get** má vstup a výstup stejný jako předchozí uživatelský požadavek.

Seznam objektů fotografií Seznam objektů fotografií uživatele lze získat pomocí požadavku `getPhotosListUser` se suffixem **list/my**. Na výstup je produkován seznam

objektů fotografií, který byl získan z databáze pomocí identifikátoru objektu uživatele.

Seznam objektů fotografií pro administrátora Verze požadavku pro administrátora `getPhotosListAdmin` se suffixem **list** vrací celý seznam objektů fotografií. Neprovádí se kontroly identifikátorů, ale probíhá kontrola role uživatele.

Ukládání objektu fotografie Ukládat objekt fotografie je uživateli umožněno pomocí požadavku `postPhotoUser` s URL suffixem **post/my**. Vstupem je objekt fotografie a ověřuje se identifikátor objektu uživatele ve fotografii s identifikátorem objektu uživatele z autentizace. Objekt fotografie je uložen do databáze a výstupem je řetězec s výsledkem operace.

Ukládání objektu fotografie pro administrátora Administrátorská verze požadavku `postPhotoAdmin` s URL suffixem **post** ukládá objekt fotografie na vstupu bez kontroly identifikátoru. Kontroluje se pouze role uživatele ve filtru. Vstupy a výstupy zůstávají beze změny.

Mazání fotografie Mazání souboru a objektu fotografie se provádí pomocí požadavku `deletePhotoUser` s URL suffixem **delete/my**. Parametrem je identifikátor objektu fotografie. Získává se objekt fotografie z databáze pro ověření identifikátoru objektu uživatele, který se porovnává s identifikátorem objektu uživatele z autentizace. Potřebná je také cesta k souboru fotografie pro smazání tohoto souboru v souborovém systému serveru. Poté je objekt fotografie smazán v databázi a výstupem je Response.

Mazání fotografie pro administrátora Požadavek pro mazání souboru a objektu fotografie administrátorem `deletePhotoAdmin` s URL suffixem **delete** je podobný. Obsahuje stejné operace, vstupy a výstupy. Neprobíhá zde kontrola identifikátorů, ale kontroluje se role uživatele pomocí filtru.

Verifikace Ověření kryptografických prvků fotografie je provedeno pomocí požadavku `verifyPhoto` s URL suffixem **verify/my**. Na vstupu je veřejný klíč uživatele a parametrem je identifikátor objektu fotografie. Z databáze se získá objekt fotografie pomocí identifikátoru objektu z parametru. Ověřuje se identifikátor objektu uživatele v objektu fotografie srovnáním s identifikátorem objektu uživatele z autentizace. Poté se vytvoří nový objekt třídy Verification pro uložení výsledků verifikace kryptografických prvků fotografie. Provedou se veškerá ověření kryptografických prvků a výsledky se uloží do objektu, který je navrácen na výstup.

Verifikace pro administrátora Administrátor má možnost provádět verifikaci libovolné fotografie pomocí požadavku `verifyPhotoAdmin` s URL suffixem **verify**. Vstupy a výstupy jsou stejné a identifikátory se nekontrolují. Kontroluje se pouze role uživatele a postup zůstavá jinak nezměněn.

Detekce a rozpoznání objektů v obraze Detekce a rozpoznání registrační značky vozidla na fotografii je prováděna požadavkem `recognizePhoto` s URL suffixem **recognize/my**. Parametrem požadavku je identifikátor objektu fotografie. Objekt fotografie je získán z databáze pomocí identifikátoru objektu fotografie z parametru. Identifikátor objektu uživatele z objektu fotografie je kontrolován porovnáním s identifikátorem objektu uživatele z autentizace. Následně je na fotografii provedena detekce a rozpoznání registrační značky vozidla pomocí JavaANPR. Vrácen je řetězec s registrační značkou vozidla.

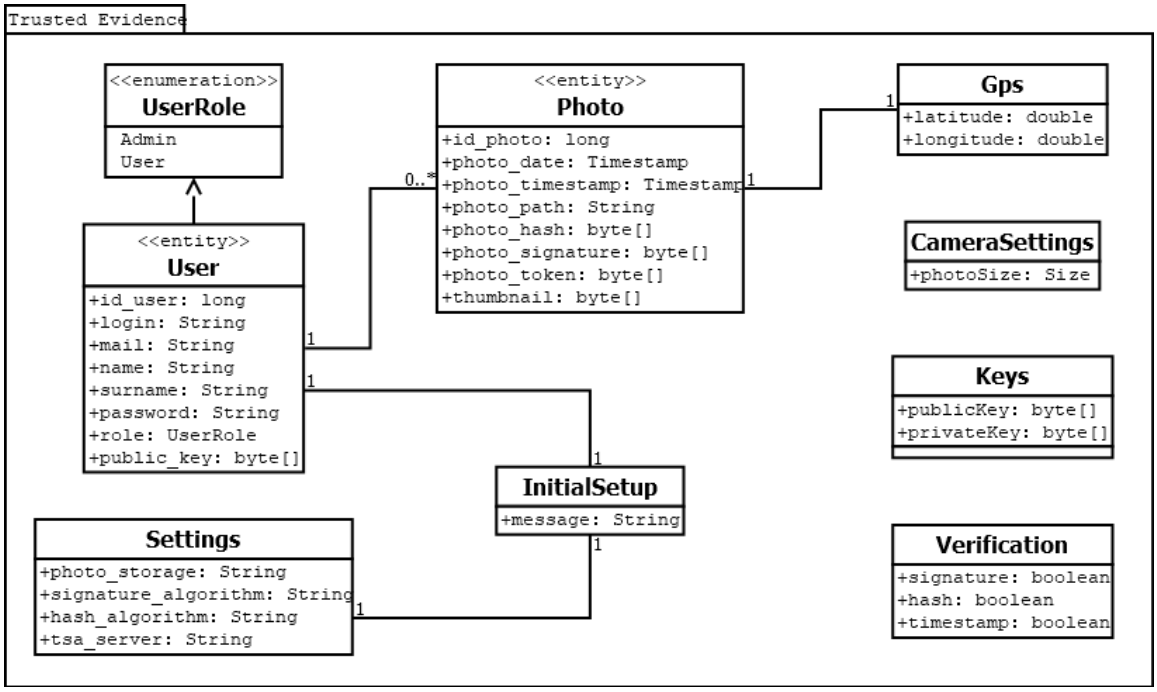
Detekce a rozpoznání objektů v obraze pro administrátora Implementována byla také verze požadavku detekce a rozpoznání objektů v obraze pro administrátora s názvem `recognizePhotoAdmin` se suffixem **recognize**. Kontrolována je pouze role uživatele bez kontroly identifikátoru objektu uživatele. Postup při provádění detekce a rozpoznání objektů v obraze je shodný s předchozím požadavkem.

4.5 Datový model

Klientská mobilní aplikace i serverová webová aplikace pracuje především s fotografiemi. Z datového hlediska bylo vhodné navrhnout a vytvořit datový model, který umožní ukládat velké množství fotografií. Důležitá však je také dostatečná rychlost a spolehlivost datových operací, aby bylo možné v rámci distribuované aplikace uspojit potřeby i většího množství uživatelů. Důraz je kladen na bezpečnost, výkonost a stabilitu výsledného databázového řešení. Zvolené řešení však respektuje typ projektu a náklady na provoz datového úložiště. Databáze na serveru používá databázový systém MySQL. Webová aplikace používá JPA, které zajišťuje objektově relační mapování. Databázové tabulky jsou tedy automaticky vytvořené z datového modelu webové aplikace. Třída datového modelu určena pro ukládání do databázového systému je označena anotací Entity. Název tabulky může být definován pomocí anotace Table s parametrem name. Identifikátor tabulky je definován anotací Id. Způsob generování hodnoty identifikátoru je definován anotací GeneratedValue pomocí parametru strategy. Unikátní sloupce v tabulce lze definovat pomocí anotace Column a parametru unique. Převedení výčtového typu na řetězec lze pomocí anotace Enumerated. Bytová pole je vhodné označit anotací Lob. Propojení tabulek pomocí cizího kliče vztahem M:1 lze provést pomocí anotace ManyToOne a JoinColumn. Pomocí parametru name definujeme název cizího klíče a pomocí parametru referencedColumnName definujeme název odkazovaného sloupce druhé tabulky. V rámci tohoto projektu budou stačit výše zmíněné koncepty. Samozřejmě lze pomocí JPA řešit i složitější problémy.

Uživatel Uživatel systému je v rámci datového modelu reprezentován třídou User anotovanou jako entita. Tabulka uživatele je anotací pojmenována jako user. Identifikátorem tabulky je identifikátor uživatele `id_user` typu long generovaný jako identita. Přihlašovací jméno `login` typu String je unikátní stejně tak jako e-mail, který je rovněž typu String. Jméno `name`, příjmení `surname` a heslo `password` jsou také typu String. Uživatelská role v systému `role` je objekt výčtu UserRole. Role je převáděna v databázi na String. UserRole je výčet, který nabývá hodnoty Admin nebo User. Miniatura profilové fotografie `thumbnail` typu byte array je uložena v databázi jako lob. Veřejný klíč uživatele `public_key` typu byte array je také anotován anotací Lob.

Fotografie, poloha a nastavení fotoaparátu Fotografie je reprezentována třídou Photo anotovanou jako entita a tabulka v databázi je pojmenována jako photo. Identifikátor tabulky je identifikátor fotografie `id_photo`, který je typu long a generován jako identita. Další vlastností třídy fotografie je uživatel `user` třídy User, který je v tabulce photo reprezentován cizím klíčem `id_user` a odkazující na `id_user` tabulky user ve vztahu M:1. Souřadnice GPS jsou uloženy pomocí vlastnosti `gps` vlastní třídy Gps. Třída Gps obsahuje hodnotu `latitude` typu double a hodnotu `longitude` typu double pro zeměpisnou šířku a délku. Časové razítko ze zařízení `photo_date` a časové razítko od důvěryhodného serveru `photo_timestamp` jsou typu Timestamp. Cesta k souboru fotografie na serveru `photo_path` je uložena jako String. Hash fotografie



Obrázek 4.10: Diagram tříd

`photo_hash`, podpis fotografie `photo_signature`, token razítka `photo_token` a miniatura `thumbnail` jsou typu byte array a uloženy jako lob. Nastavení fotoaparátu je definováno ve třídě CameraSettings, která není ukládána v databázi. Obsahuje pouze vlastnost pro rozlišení fotografie `photoSize` typu Size.

Klíče Aplikace využívá vlastní třídu Keys, která není entitou ukládanou v databází. Slouží pro uložení vygenerovaných klíčů na straně klienta a pro komunikaci se serverem. Třída obsahuje soukromý klíč `privateKey` a veřejný klíč `publicKey` typu byte array. Soukromý klíč není samozřejmě odesílán na server.

Verifikace Pro ukládání výsledku verifikace kryptografických prvků fotografie na straně serveru a pro komunikaci s klientem slouží třída Verification, která rovněž není entitou ukládanou v databázovém systému. Třída obsahuje vlastnosti `signature`, `hash` a `timestamp` typu boolean pro reprezentaci výsledku verifikace.

Nastavení od serveru Nastavení aplikace je ukládáno do objektu třídy Settings, která se nenachází v databázi. Obsahuje cestu k úložišti fotografií `photo_storage` typu String, algoritmus podpisu `signature_algorithm` typu String, algoritmus hashování `hash_algorithm` typu String a důvěryhodný server poskytující časové razítko `tsa_server` typu String. Pro počáteční nastavení od serveru byla vytvořena třída InitialSetup. Třída není entitou ukládanou v databázi. Obsahuje pouze objekt uživatele `user` třídy User, objekt nastavení `settings` třídy Settings a zprávu `message` typu String.

4.6 Nasazení a zprovoznění klientské aplikace

Zprovoznění klientské mobilní aplikace je mnohem jednodušší a rychlejší než v případě serverové webové aplikace. Klientskou aplikaci není třeba nijak složitě nasazovat. Zprovoznění na mobilním zařízení uživatele se provádí jednoduchou instalací aplikace. Aplikace je určena pouze pro mobilní zařízení s operačním systémem Android. Operační systém musí být minimálně ve verzi Android 5.0 tzv. Lollipop neboli minimálně API level 21. Pokud mobilní zařízení splňuje tyto minimální požadavky, aplikace může být na něj nainstalována a budou podporovány veškeré funkcionality. Aplikaci je možné stáhnout a nainstalovat z Google Play.

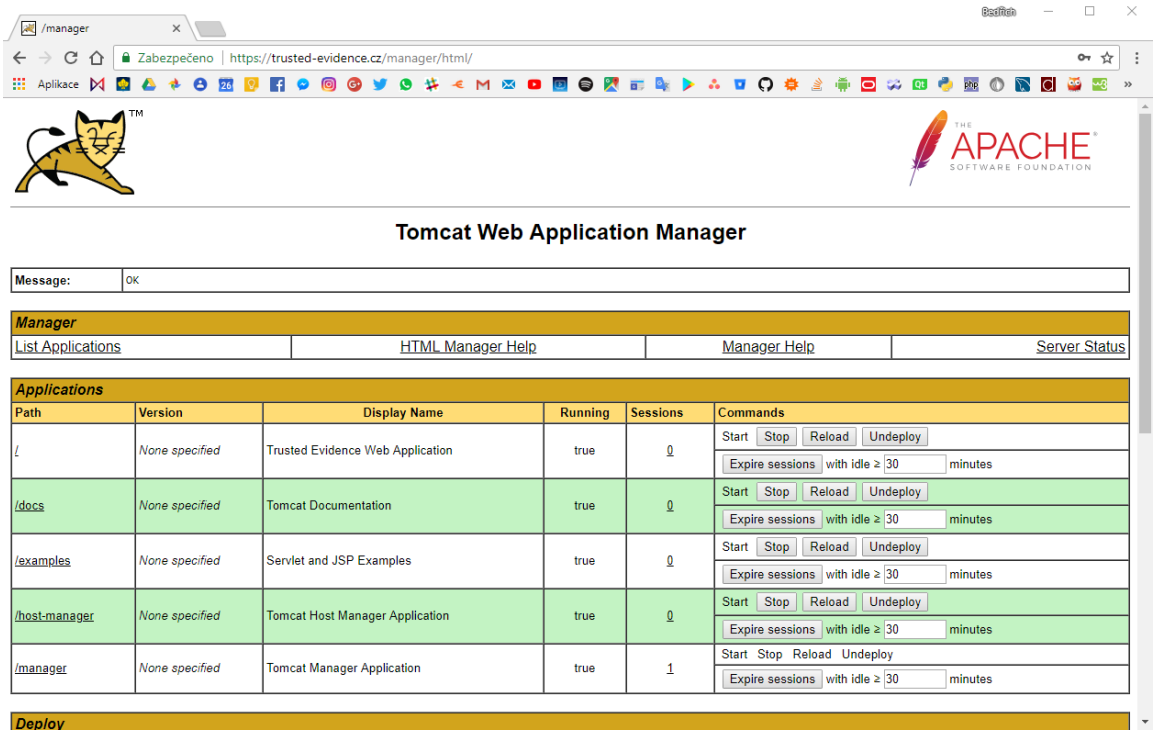
4.7 Nasazení a zprovoznění serverové aplikace

Serverová webová aplikace včetně databázového systému běží na operačním systému Ubuntu Server 16.04 LTS. Před instalací jednotlivých technologií je vhodné provést počáteční nastavení serveru. Tímto nastavením se zvýší bezpečnost serveru a jeho použitelnost. Administrativní uživatel v Linuxovém prostředí tzv. root má veškerá práva v systému a není vhodné jej používat k běžným činnostem. Změny tohoto uživatele mohou být velmi destruktivní i v případech nechtěného překlepu při zadávání příkazů. Z těchto důvodů byl proto vytvořen a nastaven alternativní uživatel s omezenějšími právy. Nový uživatel je nastaven jako superuživatel neboli běžný uživatel s právy root. Tento uživatel má možnost používat příkazy s administrativními privilegii zadáním slova sudo před samotným příkazem. Nový uživatel je proto přidán do sudo skupiny. Pro zvýšení bezpečnosti serveru je vhodné autentizovat uživatele pomocí veřejného klíče. K tomuto účelu je potřeba vygenerovat veřejný a soukromý SSH klíč. Veřejný klíč se poté kopíruje na server. Před použitím serveru je také vhodné zkontrolovat nastavení firewall. Pro použití SSH a případně dalších služeb se musí tyto služby povolit ve firewall.

Aplikační server Serverová webová aplikace implementována na platformě Java EE potřebuje pro běh aplikační server. Zvolen byl aplikační server Apache Tomcat 8. Nejprve byla na serveru zajištěna podpora pro Javu instalací OpenJDK. Pro běh služby Tomcat byl vytvořen uživatel tomcat a skupina tomcat, do které byl uživatel přidán. Uživateli byl nastaven domácí adresář v místě instalace aplikačního serveru. Uživatel byl vytvořen s parametrem shell /bin/false, takže se k tomuto účtu nemůže nikdo přihlásit. Poté byl nainstalován aplikační server Tomcat a nastaveny práva a vlastnictví pro skupinu tomcat a uživatele tomcat. Tomcat je nastaven, aby běžel jako služba. V systemd je zkontrolován adresář, kde je nainstalována Java. Ve firewall je povolen port 8080, který Tomcat využívá. Tomcat je nastaven, aby startoval automaticky po zavedení systému. V xml souboru tomcat-users byl nastaven uživatel pro přístup do webového rozhraní manažera Tomcat serveru. Pro vzdálený přístup k webovému rozhraní bylo nutné zrušit omezení na IP adresu.

Databáze Databázový systém MySQL bylo nutné pro jeho použití nainstalovat. Na čerstvé instalaci bylo vhodné spustit bezpečnostní skript, který mění nastavení databázového systému pro jeho bezpečné použití. Po úspěšné instalaci je nová instalace MySQL nakonfigurována. Instalace a konfigurace MySQL je velmi intuitivní. Vším uživatele provede instalace. Následně byla bezpečnost databázového systému ještě zvýšena provedením dalších důležitých kroků. Klíčové bylo zkontrolovat, zda v databázi není vytvořen uživatel, který má přístup k databázi bez hesla. Takové uživatele

je případně nutné zabezpečit heslem nebo odstranit. Poté již byla vytvořena databáze pro projekt. Vhodné také bylo vytvořit v databázi specifického uživatele pro aplikaci, kterému byla následně přidělena práva pouze nad vytvořenou databází. Jde o podobný princip izolace jako vytvoření superuživatele v operačním systému serveru. Snadná a intuitivní obsluha databázového systému pomocí grafického uživatelského rozhraní je možná pomocí například phpMyAdmin. Tato aplikace potřebuje pro svoji činnost PHP, které bylo nutné předem nainstalovat. Kromě samotného PHP bylo potřeba také nainstalovat potřebné PHP balíčky pro běh pod webovým serverem Apache a pro komunikaci s MySQL databází. Poté již mohla být nainstalovaná aplikace phpMyAdmin. Instalace se provádí včetně potřebných balíčků. Následně byla instalace nakonfigurována. Zvolen byl server Apache 2, jinak bylo použito výchozí nastavení konfigurace databáze a byly zadány přístupové údaje do databáze. Po provedení konfigurace bylo nutné spustit PHP rozšíření mcrypt a mbstring. Na závěr byla ještě zabezpečena aplikace phpMyAdmin, protože je častým cílem útoků. Bylo nutné ji zabezpečit před neautorizovaným přístupem. To lze snadno provést pomocí umístění brány před samotnou aplikací pomocí vestavěné funkcionality Apache pro autentizaci a autorizaci. Zapnuto bylo použití .htaccess v konfiguračním souboru phpMyAdmin v Apache přidáním AllowOverride All direktivy mezi tagy Directory phpMyAdmin. Následně byl vytvořen soubor .htaccess v adresáři phpMyAdmin, ve kterém byla nastavena autentizace. Nutné bylo vytvořit ještě soubor .htpasswd ve stejném adresáři pro autentizaci pomocí nástroje htpasswd. Po přidání uživatele je nástrojem vyžadováno zadání a potvrzení hesla. Nyní je již vše připraveno pro nasazení a použití aplikace.



Obrázek 4.11: Tomcat manager

Nasazení Pro nasazení webové aplikace na aplikačním serveru je nutné otevřít adresu webového rozhraní manažera aplikačního serveru Tomcat ve webovém prohlížeči. Uživatel se následně autentizuje zvolenými přihlašovacími údaji. Po autentizaci má uživatel k dispozici seznam běžících aplikací na aplikačním serveru. Uživatel má plnou kontrolu nad běžícími aplikacemi, které může zastavit nebo zesadit tzv. undeploy. Ve výchozím stavu na aplikačním serveru běží dokumentace a příklady, které může uživatel zesadit. Nasazení webové aplikace se provádí snadno výběrem war souboru projektu ze souborového systému uživatele a stiskem tlačítka Deploy. Aplikace po nahrání na server je okamžitě dostupná a schopná provozu. Pro běh aplikace je ještě nutné zajistit konzistentní stav databáze. Pokud je vytvářena nová čistá databáze může uživatel vše ponechat na webové aplikaci, která pomocí JPA sestaví z datového modelu všechny potřebné tabulky v databázi a provede vše potřebné za uživatele. Pokud však již existují data, která je potřeba nahrát do databáze, je nutné provést krok navíc. Stačí otevřít webové rozhraní phpMyAdmin na příslušné adrese v internetovém prohlížeči. Po provedení autentizace do phpMyAdmin lze snadno vybrat požadovanou databází a importovat databázi pomocí databázového SQL skriptu ze souborového systému uživatele.

4.8 Bezpečná výměna klíču a šifrování

Ve výchozím stavu není komunikace aplikačního serveru Tomcat a klientů šifrována. Přenášená data včetně citlivých informací a hesel mohou být snadno zneužita. Důležité je tedy zabezpečit komunikaci pomocí technologie SSL. Existuje několik způsobů, jak toho dosáhnout. Výhodné je řešení pomocí webového serveru. Zvolena byla varianta pomocí webového serveru Apache. Tomcat sice nabízí nativní podporu pro SSL, ale existuje několik nevýhod tohoto řešení. Tomcat nelze snadno navázat na konvenční SSL port. Tomcat s použitím SSL není příliš široce podporován dalším software. Například Let's Encrypt nativně nepodporuje Tomcat s SSL. Webové servery jsou častěji aktualizované než Tomcat a to přináší rychlejší záplaty a bezpečnostní aktualizace. Použitím webového serveru a reverzní proxy se lze vyhnout všem výše zmíněným problémům a výsledné řešení bude bezpečnější. Web server tak zpracovává HTTPS požadavky a předává požadavky aplikačnímu serveru Tomcat. Předávání požadavků je prováděno pomocí mod_jk modulu na webovém serveru, který komunikuje s Tomcat pomocí Apache JServ protokolu. Tomcat ve výchozím stavu má zapnutý konektor pro tento protokol a tudíž je připraven zpracovávat tyto požadavky. Nejprve bylo tedy nutné nainstalovat webový server Apache. Webový server byl povolen ve firewall. V konfiguračním souboru apache2.conf byla nastavena doména serveru. Následně již mohl být zabezpečen webový server Apache pomocí Let's Encrypt. Nainstalován byl Let's Encrpyt klient Certbot. Pomocí Certbot klienta byl vygenerován SSL certifikát pro Apache a doménu serveru. Automatické obnovování Let's Encrypt certifikátu bylo nastaveno také pomocí Certbot klienta, který každých 90 dní obnoví certifikát automaticky pomocí cron skriptu. Dále byl nainstalován mod_jk modul pro Apache. Ve workers.properties byla nastavena cesta k aplikačnímu serveru Tomcat. Důležité bylo upravit VirtualHost webového serveru Apache na proxy s použitím mod_jk. Do přislušných konfiguračních souborů dle používaných portů byl vložen JKMount ajp13_worker mezi tagy VirtualHost.

4.9 Popis fungování a použití díla

Klient Stiskem ikony Trusted Evidence je spuštěna mobilní klientská aplikace. Po spuštění aplikace se uživatel nachází na hlavní obrazovce aplikace. Tato obrazovka nabízí uživateli pod logem aplikace informace o funkcionalitě systému. Ve spodní části obrazovky se pak nachází tlačítka pro přihlášení a registraci. Po stisku tlačítka Přihlášení se uživateli zobrazí obrazovka s přihlašovacím formulářem. V tomto formuláři uživatel zadává přihlašovací jméno a heslo. Může být také nastaveno pamatování zadaných přihlašovacích údajů. Pro přihlášení slouží tlačítko Přihlásit. Uživatel je po stisku tlačítka úspěšně přihlášen, pokud jeho účet již existuje a zadal odpovídající přihlašovací údaje. Pod formulářem je uživateli nabízena možnost zaslání zapomenutého hesla. V případě zvolení této možnosti je uživateli zobrazen formulář, kde vyplňuje e-mail, na který jsou po stisku tlačítka Odeslat zaslány přihlašovací údaje k jeho účtu. Nutnou podmínkou je, aby účet uživatele již existoval a byl zadán odpovídající e-mail. V případě stisku tlačítka Registrace na hlavní obrazovce aplikace je uživateli zobrazena obrazovka s registračním formulářem. Na této obrazovce uživatel vyplňuje login, e-mail, jméno, příjmení a heslo. Uživatel je poté registrován stiskem tlačítka Registrovat. Po úspěšném přihlášení či registraci je uživateli zobrazena obrazovka galerie fotografií. Obrazovka galerie zobrazuje seznam fotografií vytvořených pomocí aplikace. Každý prvek seznamu se skládá z miniatury fotografie, názvu souboru, jména uživatele, časového razítka ze zařízení a GPS souřadnic. Navíc prvek obsahuje kontextové menu, které nabízí dostupné operace. Mezi operacemi je zobrazení fotografie, zobrazení podrobností fotografie, verifikace kryptografických prvků fotografie, detekce, rozpoznání registrační značky vozidla na fotografii a mazání fotografie. Vytvářet novou fotografii lze pomocí plovoucího tlačítka v pravém dolním rohu obrazovky. Menu aplikace se nachází ve výsuvné nabídce vlevo. Lze jej otevřít také pomocí tlačítka v levé části horní lišty. V případě stisku tlačítka fotoaparátu se zobrazí obrazovka fotoaparátu. Tuto obrazovku tvoří z větší části náhled fotografie a ve spodní části jsou ovládací tlačítka fotoaparátu. Vlevo se nachází tlačítko nastavení fotoaparátu, uprostřed tlačítko pro fotografování a vpravo tlačítko pro návrat do galerie. V případě stisku tlačítka nastavení se otevře obrazovka nastavení, kde uživatel může změnit rozlišení fotografie v rozevírací nabídce. Změny se ukládají stiskem tlačítka Uložit. V případě stisku tlačítka fotografování dochází k zachycení fotografie, vytvoření časového razítka v zařízení, uložení fotografie do úložiště mobilního zařízení, podepsání fotografie pomocí algoritmu SHA256withRSA, vytvoření SHA-256 hashe fotografie, získání GPS souřadnic a vytvoření miniatury fotografie. Fotografie včetně objektu je odeslána na server, kde je získáno časové razítko od důvěryhodného serveru. Výsledek operací je zobrazen uživateli v dialogu. V menu aplikace může uživatel také zobrazit svůj uživatelský účet výběrem možnosti Uživatel. Uživateli se zobrazí obrazovka, která obsahuje miniaturu uživatelské profilové fotografie, jméno, příjmení, login, e-mail a roli. Stiskem plovoucího tlačítka pro úpravu uživatelských údajů v pravém dolním rohu se otevře uživateli obrazovka, kde může měnit profilovou miniaturu fotografie, jméno, příjmení a e-mail. Změny jsou uloženy stiskem tlačítka Uložit. Z menu lze dále zobrazit nastavení výběrem možnosti Nastavení. Uživateli je zobrazena obrazovka s počátečním nastavením od serveru. V aplikaci lze provádět správu klíčů výběrem možnosti Klíče z menu. Je umožněno kopírovat veřejný klíč na server a provádět export či import klíčů. Poslední možností v menu aplikace je odhlášení z aplikace stiskem tlačítka Odhlášení. Po odhlášení je uživateli zobrazena opět hlavní obrazovka aplikace.

Server Webová aplikace je spuštěna zadáním adresy https://trusted-evidence.cz do webového prohlížeče. Otevřena je hlavní stránka zobrazující prezentaci obrázků klientské a serverové aplikace. Pod prezentací se nachází abstrakt práce a informace o projektu. V levé částí stránky je pak umístěno menu webové aplikace. Menu je rozděleno na panel pro veřejnou část aplikace a panel pro privátní část aplikace, která je dostupná po autentizaci. Administrátor systému má k dispozici navíc panel se správou systému. Ve veřejném panelu menu se nachází dále možnost zobrazit informace o webové aplikaci kliknutím na odkaz Webová aplikace. Uživateli se zobrazí stránka s informacemi o webové aplikaci, kde nalezne parametry webové aplikace a její funkcionalitu. Ve veřejném panelu menu je také možnost pro zobrazení informací o klientské aplikaci kliknutím na odkaz Android aplikace. Zobrazena je stránka informující o parametrech a funkcionalitách klientské aplikace. Poslední možností ve veřejném panelu menu je možnost Kontakt. Po kliknutí na tuto možnost se zobrazí stránka s kontaktem na autora práce. Autentizace ve webové aplikaci se provádí kliknutím na tlačítko Přihlášení v horní liště nebo v privátním panelu menu. Zobrazena je stránka s přihlašovacím formulářem, kde uživatel vyplňuje login a heslo. Přihlašuje se kliknutím na tlačítko Přihlásit. Registrace uživatele se provádí kliknutím na tlačítko Registrace v horní liště nebo v privátním panelu menu. Otevřena je stránka s registračním formulářem, kde se zadává jméno, příjmení, login, e-mail a heslo. Po úspěšné registraci je uživatel odkázán na přihlášení. Po úspěšném přihlášení se uživateli zobrazí stránka Dashboard se statistikami systému z privátního panelu. V privátním panelu menu se kromě možnosti Dashboard nachází také možnost Fotografie. Po kliknutí na tuto možnost se uživateli zobrazí stránka se seznamem fotografií. Prvek seznamu obsahuje náhled fotografie, GPS souřadnice, časové razítko a dostupné operace. Mezi operacemi je zobrazení fotografie, zobrazení podrobností fotografie, verifikace kryptografických prvků fotografie, detekce, rozpoznání registrační značky vozidla na fotografii a mazání fotografie. Po kliknutí na možnost zobrazení fotografie je zobrazena zvolená fotografie. V případě kliknutí na možnost zobrazení podrobností fotografie jsou zobrazeny podrobnosti této fotografie. Možnost verifikace provede po kliknutí verifikaci všech kryptografických prvků fotografie a zobrazí výsledek uživateli. Možnost rozpoznání provede detekci a rozpoznání registrační značky vozidla na fotografii a zobrazí výsledek uživateli. Uživatel s rolí administrátora v systému má navíc v menu panel Správa systému. V tomto panelu se nachází možnost Uživatelé, která po kliknutí zobrazí seznam všech uživatelů systému. Každý prvek seznamu lze upravit nebo smazat. Úprava se provádí kliknutím na možnost Upravit. Na této stráce je formulář s vlastnostmi uživatele, které lze upravit. Změny jsou uloženy kliknutím na tlačítko Uložit. V seznamu uživatelů je také možnost vytvořit nového uživatele kliknutím na tlačítko Vytvořit nového uživatele. Po kliknutí na možnost je otevřena stránka s formulářem pro zadání údajů uživatele. Uživatel je vytvořen stiskem tlačítka Uložit. Odhlášení z webové aplikace je možné provést stiskem tlačítka Odhlásit v horní liště.

4.10 Metodika a ověření správnosti a funkčnosti díla

Práce je ověřena a hodnocena z hlediska správnosti a funkčnosti dle několika kritérií. Ověření je zaměřeno na systém jako celek a na jednotlivé prvky systému či funkcionality. Důležité bylo zaměřit se na ověření komunikace jednotlivých komponent systému. Byly porovnány předpokládané a získané výstupy testováním se zaměřením také na neočekávané a nevalidní vstupy. Systém byl posouzen z hlediska bezpečnosti a zachování integrity uži-

vatelských dat. Simulovány byly útoky na jednotlivé komponenty systému. Byly provedeny pokusy o únik dat, ovládnutí služby a znepřístupnění aplikace pro ostatní uživatele. Především bylo ověřeno, zda práce odpovídá zadání a splňuje všechny jeho definované body. Pro ověření práce byl vytvořen a proveden uživatelský experiment.

Klientská aplikace Klientská mobilní aplikace na platformě Android splňuje požadavky material designu společnosti Google. Používá moderní prvky grafického uživatelského rozhraní. Téměř ve všech případech funguje asynchronně. Umožňuje provádět operace na pozadí, kdy je aplikace nadále dostupná uživateli. Funguje rychle a nezasekává se. Na testovaných zařízeních nezpůsobuje pády a uníky dat tzv. memory leaky. Pouze přepínání fotoaparátu, přechod na jinou aktivitu z fotoaparátu a návrat do aktivity fotoaparátu způsobuje přibližně půl vteřiny zamrznutí náhledu tzv. preview fotoaparátu. Za toto chování je zodpovědné API a řešeno bývá většinou pouze zakrytím animací s černou obrazovkou.

Serverová aplikace Serverová webová aplikace na platformě Java EE taktéž splňuje požadavky moderní webové aplikace. Webová aplikace je responzivní a zobrazitelná na různých zařízeních s různým rozlišením. Podporována jsou samozřejmě také mobilní zařízení. Aplikace funguje rychle bez žádných zasekávání rozhraní. Webová aplikace umožňuje navíc po autentizaci zobrazit uživatelská data stejně jako v mobilní klientské aplikaci. Databáze a úložiště fotografií jsou vyřešeny z hlediska typu projektu a nákladů velmi dobře a nezpůsobují problémy či větší odezvy při načítání dat. Délka doby ukládání dat je závislá na nastavené velikosti rozlišení fotografie. Načítání seznamu je také závislé na počtu fotografií, protože každý záznam fotografie v databázi obsahuje miniaturu fotografie. Miniatury jsou vytvářeny velmi efektivně s použitím komprese a datově nenáročného modelu. I při velkém množství fotografií tak není doba stahovnání příliš dlouhá, ale s přibývajícím množstvím fotografií se pomalu zvyšuje.

API RESTful API vytvořené a běžící na straně serveru bylo testováno použitím programu Postman, který generuje HTTP resp. HTTPS požadavky. Odpověď na většinu požadavků byla přijata od 50ms do 300ms. Požadavek na stažení seznamu fotografií je závislý na počtu fotografií. Nahrání fotografie na server je závislé na velikosti rozlišení fotografie. Požadavky na API jsou na straně serveru zpracovávány asynchronně. Server je tedy schopen současně zpracovat velké množství požadavků. Během testování se nepodařilo API přehltit požadavky.

Kryptografické prvky Platnost všech kryptografických prvků je ověřována na serveru. Pro generování podpisu a hashe byla použita knihovna java.security, která je považována za bezpečnou. Digitální podpis je vytvářen pomocí algoritmu SHA256withRSA. Digitální podpis je ověřován pomocí veřejného RSA klíče, původní fotografie a vytvořeného podpisu. Hash je vytvářen pomocí algoritmu SHA-256. Hash se ověřuje vytvořením hashe z původní fotografie a porovnáním s existujícím hashem. Časové razítko od důvěryhodného serveru je ověřováno porovnáním hashe původní fotografie s tokenem od TSA serveru. Jako důvěryhodný server je používán freetsa.org.

Zprovoznění Ověřováno a testováno bylo také nasazení a zprovoznění klientské i serverové aplikace. Postup pro nasazení a zprovoznění všech komponent systému funguje a odpovídá postupu popsanému v této práci.

Experiment Pro ověření klient-server aplikace byl vytvořen uživatelský experiment. Tento experiment zahrnuje testování klient-server aplikace vybraným vzorkem uživatelů a

následné odpovězení na sadu otázek. Výsledky dotazníků byly poté statisticky zpracovány a vyhodnoceny. Výsledek experimentu poskytl zajímavou zpětnou vazbu a uživatelské hodnocení. Především však experiment ověřil funkčnost klient-server aplikace. První fáze experimentu spočívala v testování aplikace uživatelem. Testování zahrnuje vyzkoušení všech případů užití aplikace resp. provedení všech dostupných operací. Druhá fáze experimentu zahrnovala vyplnění dotazníku. Dotazník byl sestaven ze dvou sad otázek. První část otázek byla založena na odpovědi procentuálním hodnocením. Respondent hodnotil použitelnost, stabilitu, uživatelské rozhraní a udělil aplikaci celkové hodnocení. Druhá část otázek byla typu odpovědi ano/ne. Respondent odpovídal, zda by potřeboval nebo využil tuto aplikaci, zda by si nainstaloval aplikaci a zda by tuto aplikaci doporučil.

Zadání Zadání diplomové práce bylo splněno. Autor byl seznámen s metodami tvorby klient-server aplikací s mobilními klienty. Byla vytvořena koncepce vhodné klient-server aplikace s mobilními klienty, která pracuje s obrazovými daty a ověřuje zdroj a čas pořízení takových dat. Byly diskutovány vlastnosti vytvořené koncepce a rozebrány její výhody a nevýhody. Poté byla klient-server aplikace implementována a její funkčnost demonstrována na vhodném příkladě. Dosažené výsledky a možnosti pokračování práce jsou diskutovány v následující části práce.

4.11 Interpretace výsledků

V rámci této práce byla vytvořena klientská mobilní aplikace na platformě Android, která umožňuje bezpečně vytvářet fotografie pomocí fotoaparátu mobilního zařízení. Pro fotografii vytváří časové razítko, digitální podpis pomocí algoritmu SHA256withRSA, hash pomocí algoritmu SHA-256, miniaturu fotografie a získává GPS souřadnice. Fotografii ukládá v zařízení včetně podpisu. Objekt fotografie s časovým razítkem ze zařízení, podpisem, hashem, GPS souřadnicemi, objektem uživatele, miniaturou a původní fotografií klient odesílá na server. Na serverové webové aplikaci běží RESTful API a databáze. Webová aplikace získává časové razítko od důvěryhodného serveru. Serverová aplikace umožňuje verifikovat všechny kryptografické prvky včetně časového razítka. Serverová aplikace využívá důvěryhodný TSA server freetsa.org. Webová aplikace detekuje a rozpoznává registrační značku vozidla z uložené fotografie. Tato aplikace se stará o správu veškerých dat a bezpečnost před útoky. Klientská aplikace je dostupná na Google Play. Serverová webová aplikace je nasazena a běží na adrese https://trusted-evidence.cz. Webová aplikace umožňuje administrátorovi spravovat systém a běžnému uživateli prohlížet data.

Kapitola 5

Závěr

Cílem práce bylo vytvořit klient-server aplikaci s mobilními klienty, která pracuje s obrazovými daty a ověřuje zdroj a čas pořízení dat. Záměr práce byl splněn. Bylo úspěšně dosaženo požadovaného cíle. Jednotlivé body zadání byly splněny.

Seznámil jsem se s metodami tvorby klient-server aplikací s mobilními klienty. Provedl jsem analýzu a rešerši aktuálně používaných technologií pro vývoj klient-server aplikací. Prošel jsem statistiky používanosti jednotlivých řešení. Zhodnotil jsem výhody a nevýhody jednotlivých řešení a vhodnost použití pro tento typ projektu. Použil jsem nejpoužívanější klientskou mobilní platformu a kvalitní serverové řešení. Byla vytvořena koncepce vhodné klient-server aplikace s mobilními klienty, která pracuje s obrazovými daty a ověřuje zdroj a čas pořízení takových dat. Vytvořil jsem koncept klientské aplikace, která bezpečně pořizuje fotografii, vytváří digitální podpis, časové razítko, hash a získává GPS souřadnice. Server zajišťuje časové razítko od důvěryhodného serveru, provádí verifikace kryptografických prvků, detekci a rozpoznání objektů. Byly diskutovány vlastnosti vytvořené koncepce a rozebrány její výhody a nevýhody. Výhodou je efektivní, distribuované, energeticky a datově nenáročné řešení. Klient vykonává pouze nezbytné operace a zbytek je prováděn na serveru. Klient-server aplikace byla implementována a její funkčnost demonstrována na vhodném příkladě. Klientská aplikace byla implementována a je dostupná na Google Play. Webová aplikace je nasazená a dostupná na virtuálním privátním serveru. Klient-server aplikace byla otestována a ověřena. Byl proveden uživatelský experiment.

Klient-server aplikace byla v rámci uživatelského experimentu testována 15 respondenty. Aplikaci by potřebovalo nebo využilo 86.7% respondentů. Aplikaci by nainstalovalo 73.3% respondentů a dále doporučilo 93.3% respondentů. Uživatelé vyhodnotili použitelnost aplikace na 92%, stabilitu aplikace na 99% a uživatelské rozhraní na 91%. Aplikace obdržela celkové hodnocení 93%.

Díky této práci jsem se zlepšil v návrhu a implementaci klient-server aplikací. Získal jsem zkušenosti při tvorbě front-end i back-end částí mobilní a webové aplikace. Práce mě naučila tvořit komunikaci klientské a serverové aplikace pomocí RESTful API z obou stran. Cennou zkušeností bylo také provádění kryptografických operací a zpracování obrazu v rámci klient-server aplikace. Získal jsem přehled v oblasti důvěryhodného dokumentu a jeho použití jako důkazního materiálu. Práce mě hodně bavila a rád bych v ní pokračoval.

Na této práci lze pokračovat, rozšiřovat ji a nadále zvyšovat její kvalitu. Vhodné je přidat podporu pro další typy dokumentů např. video. Lze vylepšit ukládání dat, přidat cachování a šifrování dat v databázi. Přínosem by bylo cloud úložiště pro ukládání fotografií.

Literatura

[1] *Zákon č. 227/2000 Sb. o elektronickém podpisu a o změně některých dalších zákonů (zákon o elektronickém podpisu)*. Červen 2000, [Online; navštíveno 23.11.2017].
URL https://www.zakonyprolidi.cz/cs/2000-227

[2] *Zákon č. 235/2004 Sb. o dani z přidané hodnoty*. Duben 2004, [Online; navštíveno 23.11.2017].
URL https://portal.gov.cz/app/zakony/zakonPar.jsp?idBiblio=57849&nr=235~2F2004&rpp=15#local-content

[3] *Zákon č. 499/2004 Sb. o archivnictví a spisové službě a o změně některých zákonů*. Červenec 2004, [Online; navštíveno 23.11.2017].
URL https://portal.gov.cz/app/zakony/zakonPar.jsp?idBiblio=58364&nr=499~2F2004&rpp=15#local-content

[4] *Nařízení Evropského parlamentu a Rady (EU) č. 910/2014 ze dne 23. července 2014 o elektronické identifikaci a službách vytvářejících důvěru pro elektronické transakce na vnitřním trhu a o zrušení směrnice 1999/93/ES)*. Srpen 2014, [Online; navštíveno 23.11.2017].
URL http://data.europa.eu/eli/reg/2014/910/oj

[5] *Zákon č. 297/2016 Sb. o službách vytvářejících důvěru pro elektronické transakce*. Srpen 2016, [Online; navštíveno 23.11.2017].
URL https://portal.gov.cz/app/zakony/zakonPar.jsp?idBiblio=87076&nr=297~2F2016&rpp=15#local-content

[6] *Zákon č. 298/2016 Sb., kterým se mění některé zákony v souvislosti s přijetím zákona o službách vytvářejících důvěru pro elektronické transakce, zákon č. 106/1999 Sb., o svobodném přístupu k informacím, ve znění pozdějších předpisů, a zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů*. Srpen 2016, [Online; navštíveno 23.11.2017].
URL https://portal.gov.cz/app/zakony/zakonPar.jsp?idBiblio=87077&nr=298~2F2016&rpp=15#local-content

[7] *Zákon č. 250/2017 Sb. o elektronické identifikaci*. Červenec 2017, [Online; navštíveno 23.11.2017].
URL https://www.zakonyprolidi.cz/cs/2017-250

[8] Adams, C.; Cain, P.; Pinkas, D.; aj.: *RFC 3161 - Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)*. Srpen 2001, [Online; navštíveno

24.11.2017].
URL https://tools.ietf.org/html/rfc3161

[9] Antipolis, S.: *ETSI publishes European Standards to support eIDAS regulation.* Červenec 2016, [Online; navštíveno 24.11.2017].
URL http://www.etsi.org/news-events/news/1111-2016-07-etsi-publishes-european-standards-to-support-eidas-regulation

[10] Benatallah, B.; Casati, F.; Toumani, F.: *Web service conversation modeling: A cornerstone for e-business automation.* 2004, [Online; navštíveno 27.12.2017].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.132.7764&rep=rep1&type=pdf

[11] Berners-Lee, T.; Fielding, R.; Masinter, L.: *Uniform Resource Identifier (URI): Generic Syntax. RFC3986.* Leden 2005, [Online; navštíveno 2.1.2018].
URL https://tools.ietf.org/html/rfc3986

[12] Birrell, A. D.; Nelson, B. J.: *Implementing Remote Procedure Calls. Xerox Palo Alto Research Center.* [Online; navštíveno 30.12.2017].
URL http://www.cs.cmu.edu/~dga/15-712/F07/papers/birrell842.pdf

[13] Carpenter, B.; Cheshire, S.; Hinden, R.: *Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers. RFC6874.* Únor 2013, [Online; navštíveno 2.1.2018].
URL https://tools.ietf.org/html/rfc6874

[14] Comer, D.: *Internetworking With TCP/IP Vol 1: Principles, Protocols, and Architecture.* 2000, [Online; navštíveno 27.12.2017].
URL http://cpe.rmutt.ac.th/network/images/cn/[3]Comer_Douglas_Internetworking_with_TCP_IP_Vol.1.pdf

[15] ETSI: *Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES).* Duben 2013, [Online; navštíveno 25.11.2017].
URL http://www.etsi.org/deliver/etsi_ts/101700_101799/101733/02.02.01_60/ts_101733v020201p.pdf

[16] ETSI: *Electronic Signatures and Infrastructures Activities.* Říjen 2017, [Online; navštíveno 24.11.2017].
URL https://portal.etsi.org//TBSiteMap/ESI/ESIActivities.aspx

[17] Fielding, R.; Gettys, J.; Mogul, J.; aj.: *Hypertext Transfer Protocol – HTTP/1.1. RFC2068.* Leden 1997, [Online; navštíveno 30.12.2017].
URL https://tools.ietf.org/html/rfc2068

[18] Fielding, R.; Gettys, J.; Mogul, J.; aj.: *Hypertext Transfer Protocol – HTTP/1.1. RFC2616.* Červen 1999, [Online; navštíveno 30.12.2017].
URL https://tools.ietf.org/html/rfc2616

[19] Fielding, R.; Reschke, J.: *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. RFC7230.* Červen 2014, [Online; navštíveno 2.1.2018].
URL https://tools.ietf.org/html/rfc7230

[20] Fielding, R.; Reschke, J.: *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC7231.* Červen 2014, [Online; navštíveno 2.1.2018].
URL https://tools.ietf.org/html/rfc7231

[21] Held, G.: *Server Management.* Březen 2000.

[22] Hloušková, V.; Lubas, J.; Rosol, I.; aj.: *Důvěryhodný digitální dokument - Stanovisko ICT UNIE k problematice právně validního dokumentu.* Březen 2014, [Online; navštíveno 23.11.2017].
URL http://www.ipsd.cz/wp-content/uploads/2017/03/2014-03-25-ictu-duveryhodny-dokument-final-12-ds.pdf

[23] Hock-Chuan, C.: *HTTP Basics.* [Online; navštíveno 13.5.2018].
URL https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

[24] Hohpe, G.: *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.* Březen 2012.

[25] Housley, R.: *RFC 5083 - Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type.* Listopad 2007, [Online; navštíveno 24.11.2017].
URL https://tools.ietf.org/html/rfc5083

[26] IBM: *IBM Operating System/360 Concepts and Facilities.* 1965, [Online; navštíveno 27.12.2017].
URL http://www.bitsavers.org/pdf/ibm/360/os/R01-08/C28-6535-0_OS360_Concepts_and_Facilities_1965.pdf

[27] Israel, J.; J. Mitchell, H. S.: *Separating Data from Function in a distributed File System. Xerox Palo Alto Research Center.* 1978, [Online; navštíveno 27.12.2017].
URL https://priorart.ip.com/IPCOM/000128883

[28] Microsoft: *ASP.NET.* [Online; navštíveno 29.12.2017].
URL https://www.asp.net/

[29] Microsoft: *ASP.NET Web Application Project Deployment Overview.* [Online; navštíveno 29.12.2017].
URL https://msdn.microsoft.com/en-us/library/dd394698%28VS.100%29.aspx

[30] Microsoft: *Common Web Application Architectures.* [Online; navštíveno 13.5.2018].
URL https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/common-web-application-architectures

[31] Microsoft: *How RPC Works.* [Online; navštíveno 30.12.2017].
URL https://technet.microsoft.com/cs-cz/library/cc738291(v=ws.10).aspx

[32] NIST: *FIPS 186-4 - Digital Signature Standard (DSS).* Červenec 2013, [Online; navštíveno 25.11.2017].
URL http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[33] Oracle: *Java RMI Specification.* [Online; navštíveno 30.12.2017].
URL https://docs.oracle.com/javase/8/docs/platform/rmi/spec/rmiTOC.html

[34] Oracle: *The Java EE 7 Tutorial.* [Online; navštíveno 13.5.2018].
URL https://docs.oracle.com/javaee/7/JEETT.pdf

[35] Oracle: *The Java Remote Method Invocation API (Java RMI).* [Online; navštíveno 30.12.2017].
URL https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/

[36] Pinkas, D.; Pope, N.; Ross, J.: *RFC 5126 - CMS Advanced Electronic Signatures (CAdES).* Únor 2008, [Online; navštíveno 24.11.2017].
URL https://tools.ietf.org/html/rfc5126

[37] Richardson, L.; Amundsen, M.; Ruby, S.: *RESTful Web APIs.* Září 2013.

[38] Rivest, R.; Shamir, A.; Adleman, L.: *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* 1978, [Online; navštíveno 25.11.2017].
URL http://people.csail.mit.edu/rivest/Rsapaper.pdf

[39] Rulifson, J.: *DEL. IETF. RFC5.* Červen 1969, [Online; navštíveno 27.12.2017].
URL https://tools.ietf.org/html/rfc5

[40] Santesson, S.; Pope, N.: *RFC 5816 - ESSCertIDv2 Update for RFC 3161.* Březen 2010, [Online; navštíveno 24.11.2017].
URL https://tools.ietf.org/html/rfc5816

[41] Shapiro, E.: *Network Timetable. IETF. RFC4.* Březen 1969, [Online; navštíveno 27.12.2017].
URL https://tools.ietf.org/html/rfc4

[42] Statcounter.com: *Mobile Operating System Market Share Worldwide. Q1 - Q4 2017.* [Online; navštíveno 28.12.2017].
URL http://gs.statcounter.com/os-market-share/mobile/worldwide/#quarterly-201701-201704

[43] Statista.com: *Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2017.* [Online; navštíveno 28.12.2017].
URL https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

[44] Sun Microsystems, I.: *RPC: Remote Procedure Call Protocol Specification Version 2. RFC1057.* [Online; navštíveno 30.12.2017].
URL https://tools.ietf.org/html/rfc1057

[45] W3techs.com: *Usage of web servers for websites.* [Online; navštíveno 13.5.2018].
URL https://w3techs.com/technologies/overview/web_server/all

[46] Wikipedia: *Trusted timestamping.* [Online; navštíveno 14.5.2018].
URL https://en.wikipedia.org/wiki/Trusted_timestamping

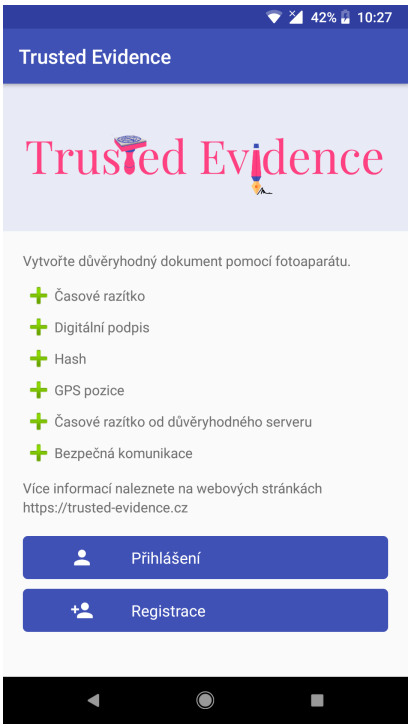
Příloha A

Obsah přiloženého paměťového média

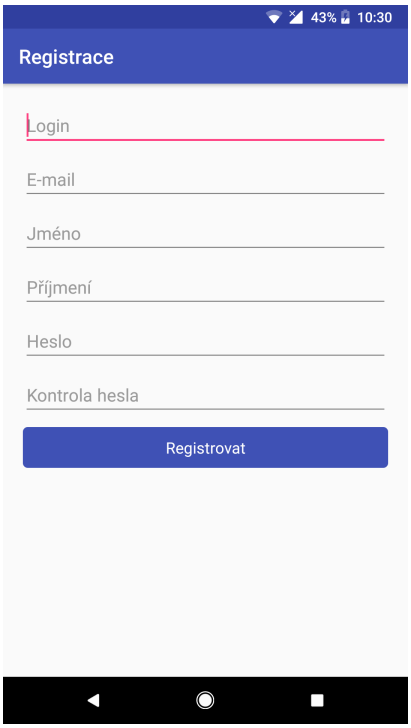
/client/	Zdrojový kód klientské aplikace
/server/	Zdrojový kód serverové aplikace
/text/	Text práce a zdrojový kód textu
/LICENSE	Licence
/README	Návod k použití

Příloha B

Snímky klientské aplikace

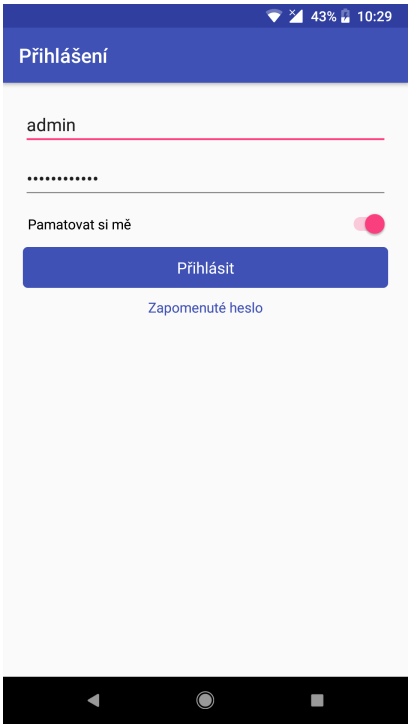


(a) Hlavní obrazovka

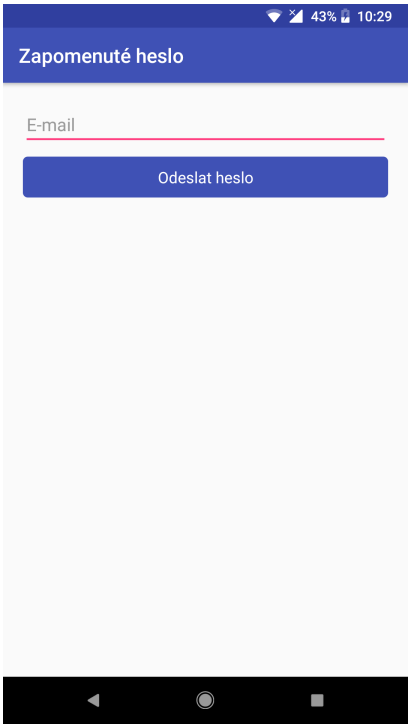


(b) Registrace

Obrázek B.1: Hlavní obrazovka a registrace

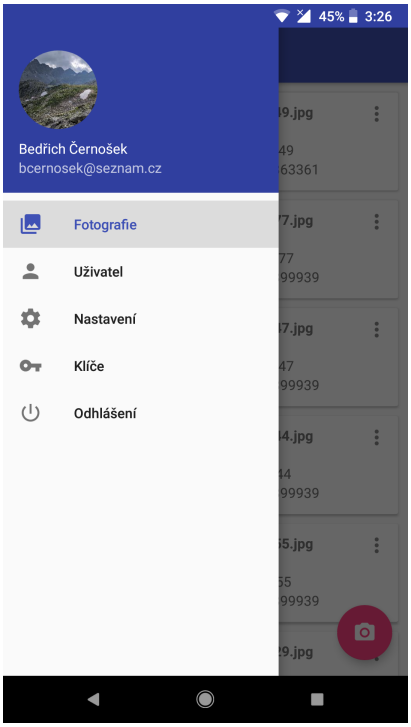


(a) Přihlášení

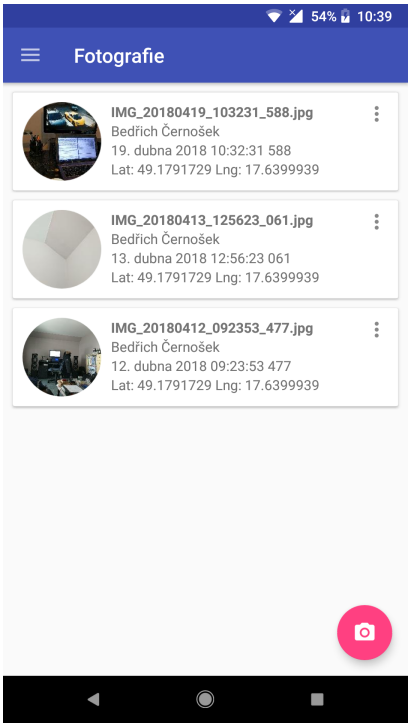


(b) Zapomenuté heslo

Obrázek B.2: Přihlášení a zapomenuté heslo

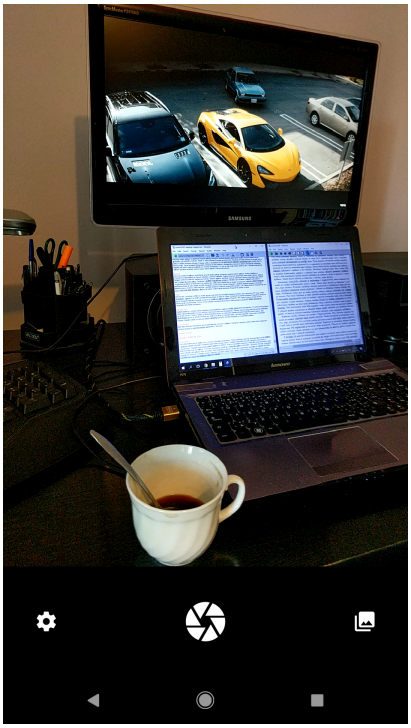


(a) Menu

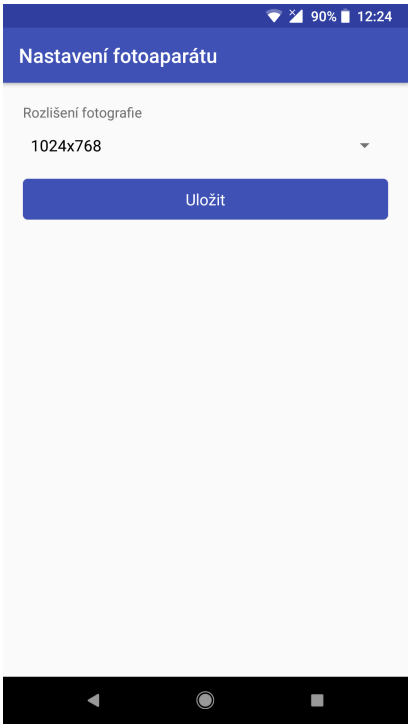


(b) Galerie

Obrázek B.3: Menu a galerie

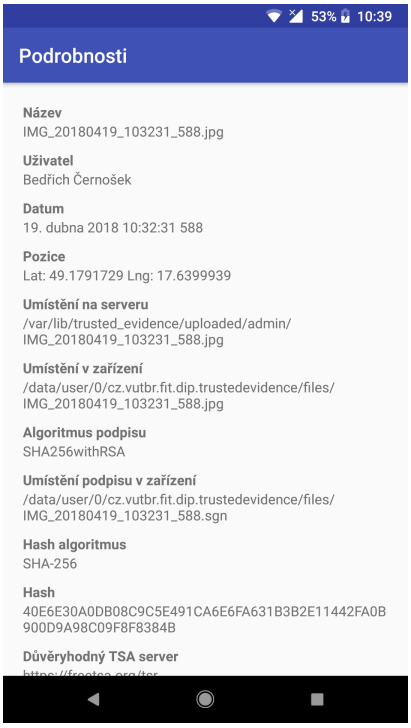


(a) Fotoaparát

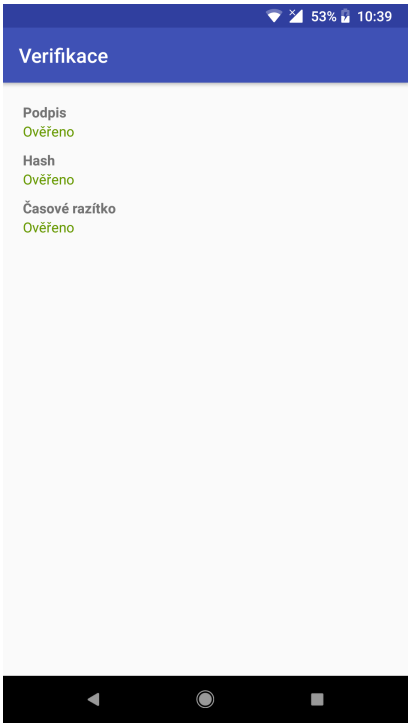


(b) Nastavení fotoaparátu

Obrázek B.4: Fotoaparát a jeho nastavení

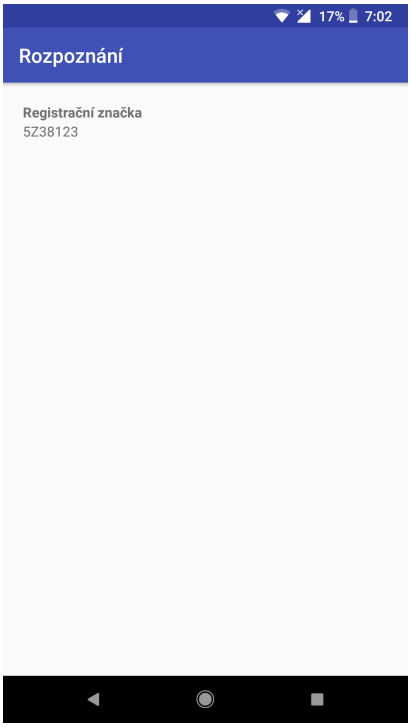


(a) Podrobnosti fotografie

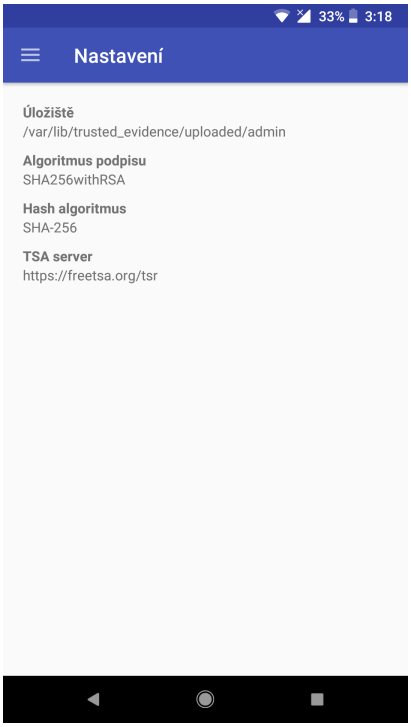


(b) Verifikace kryptografických prvků

Obrázek B.5: Podrobnosti a verifikace kryptografických prvků fotografie

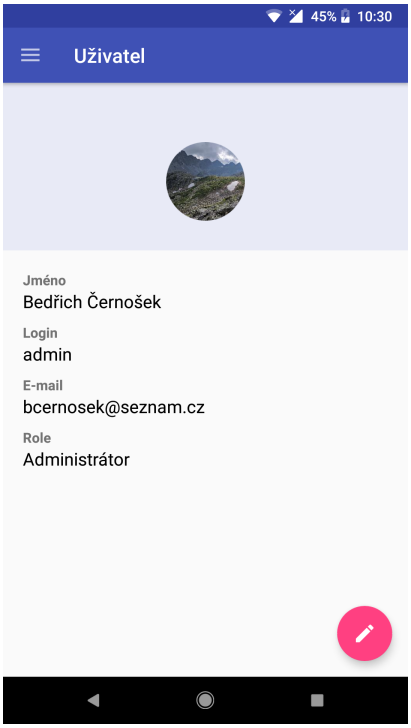


(a) Rozpoznání registrační značky

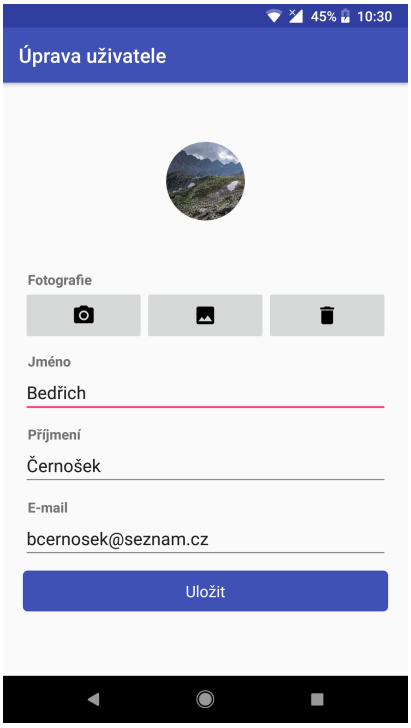


(b) Nastavení

Obrázek B.6: Rozpoznání registrační značky vozidla a počáteční nastavení od serveru



(a) Uživatel

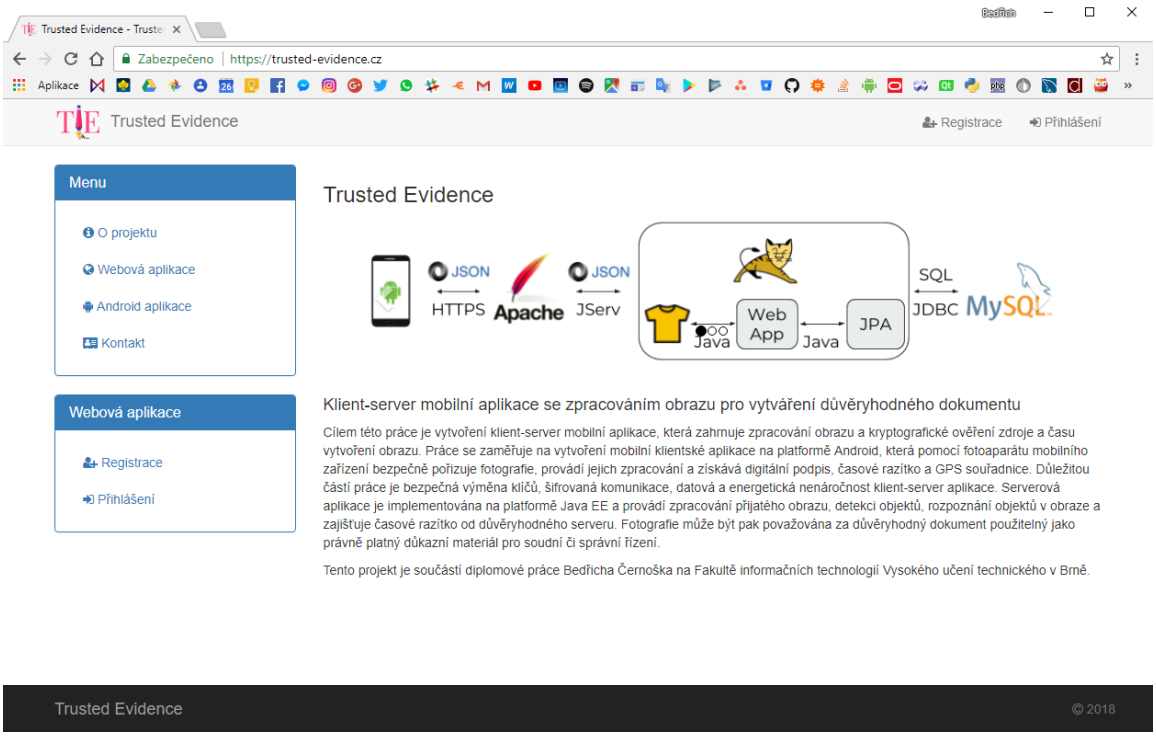


(b) Úprava uživatele

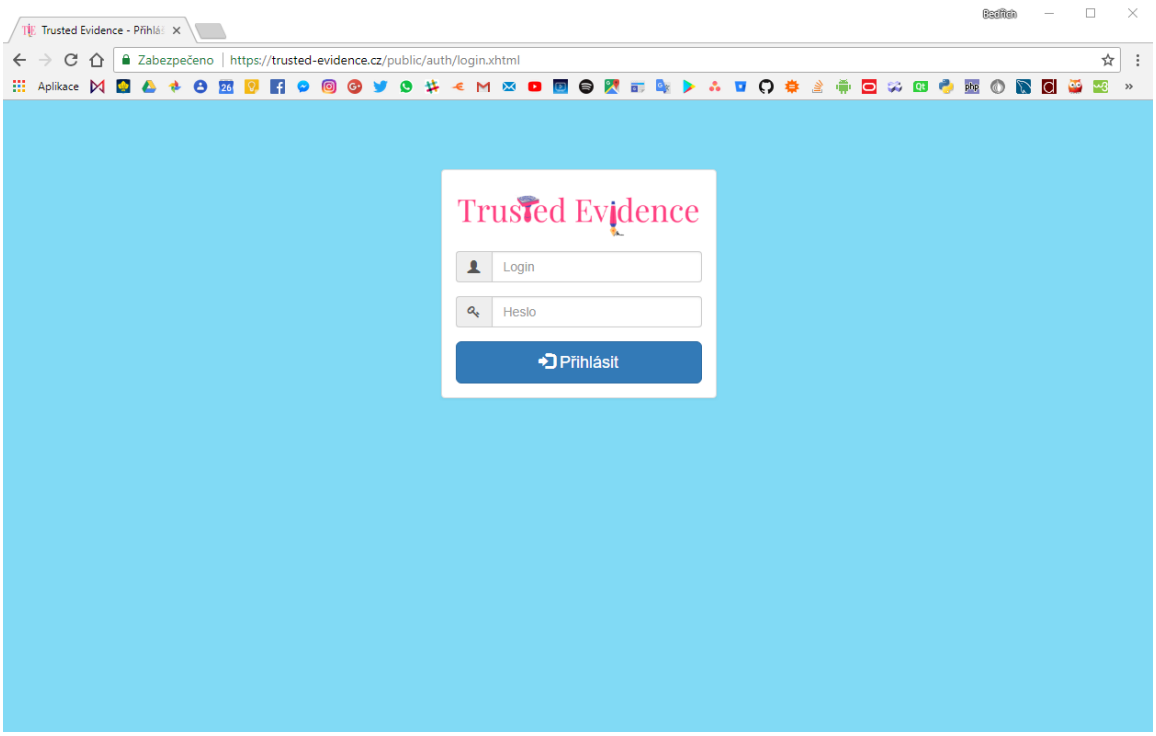
Obrázek B.7: Uživatel a úprava uživatele

Příloha C

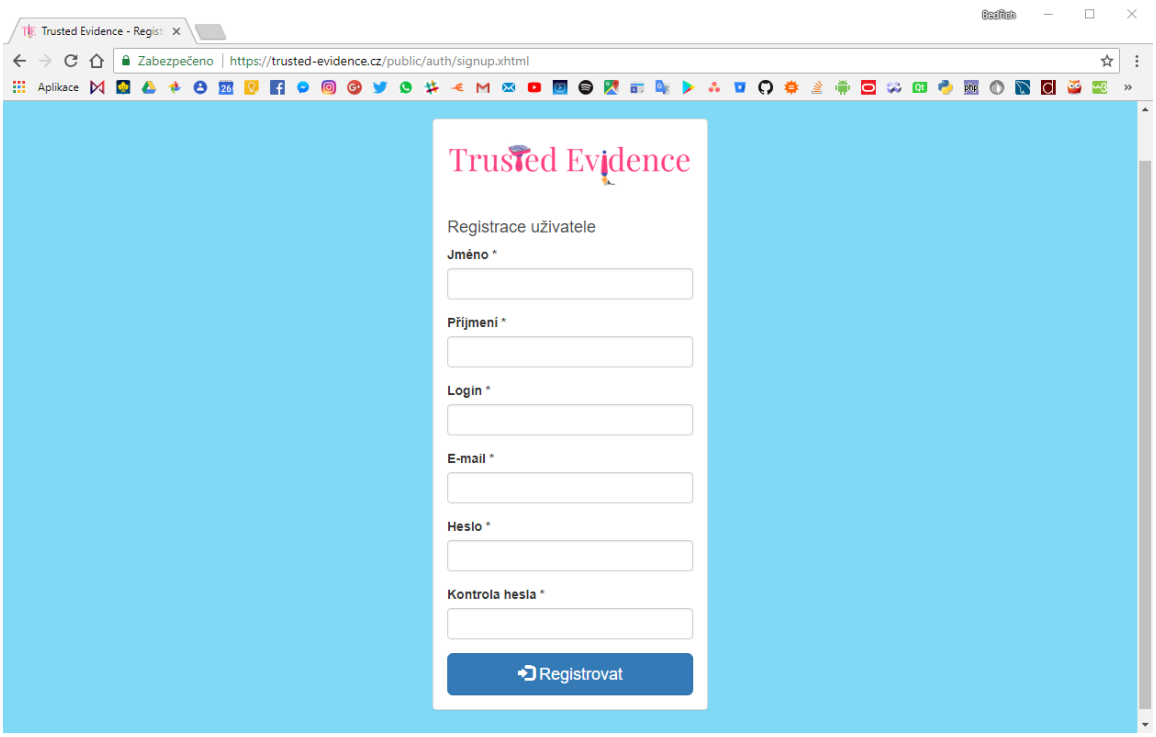
Snímky serverové aplikace



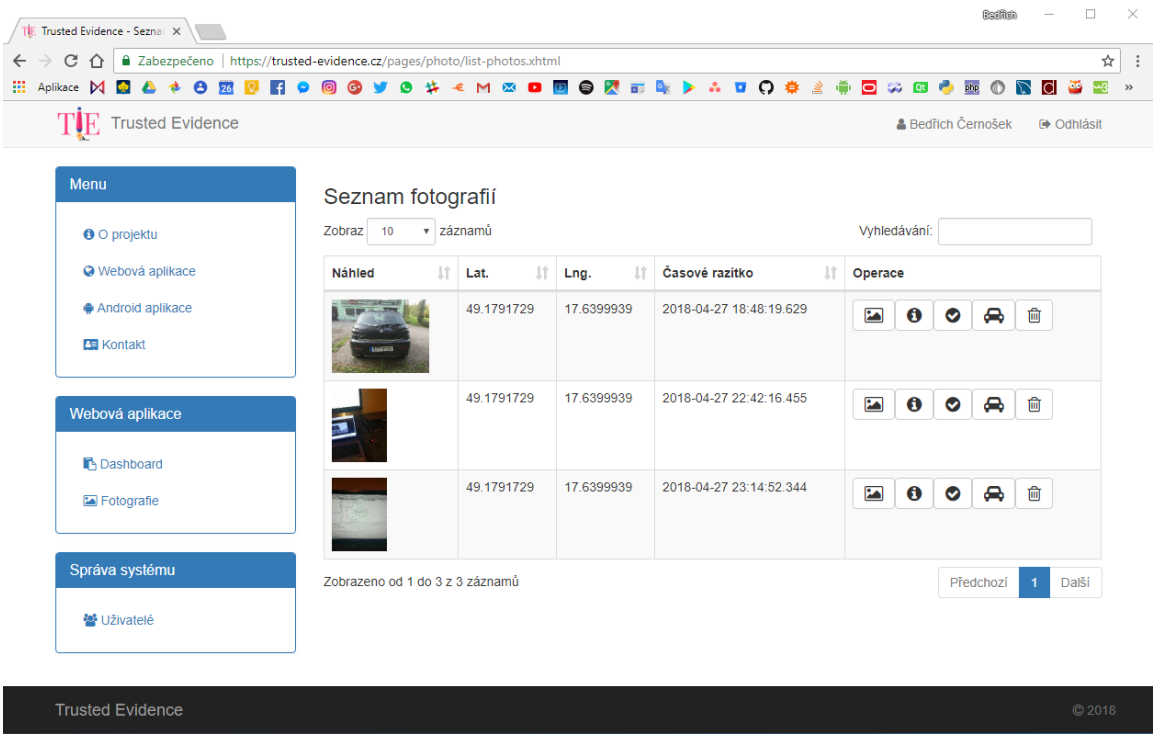
Obrázek C.1: Hlavní stránka



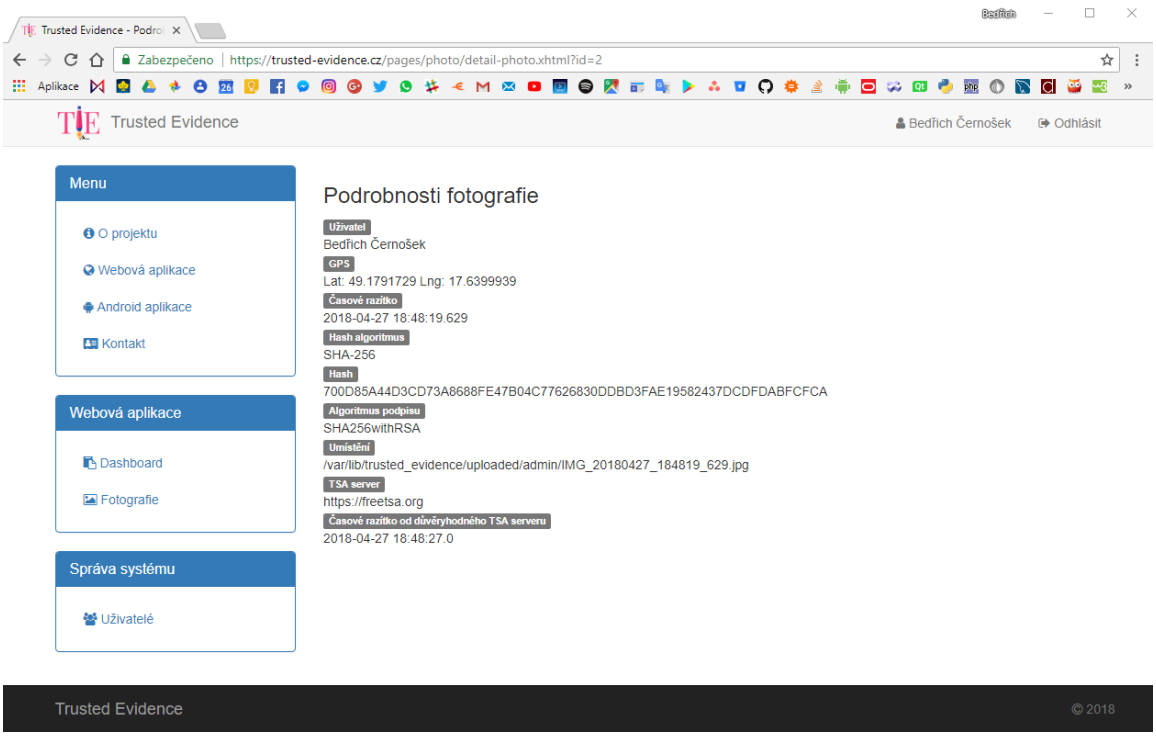
Obrázek C.2: Přihlášení



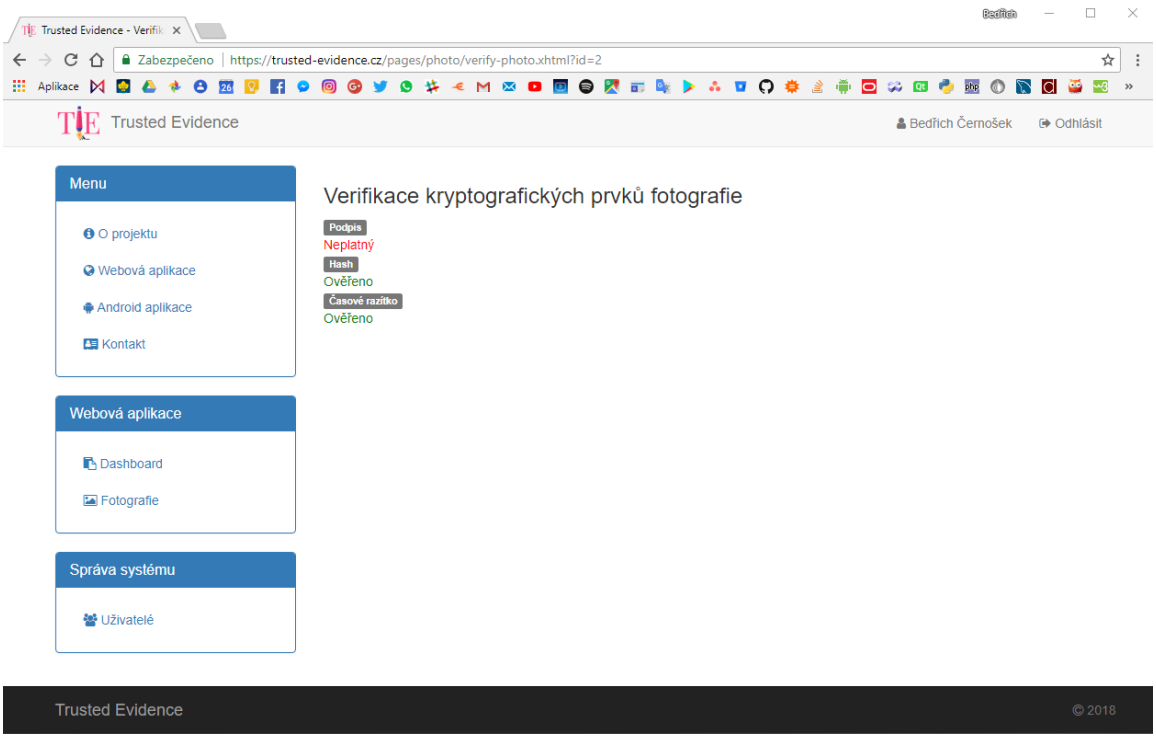
Obrázek C.3: Registrace



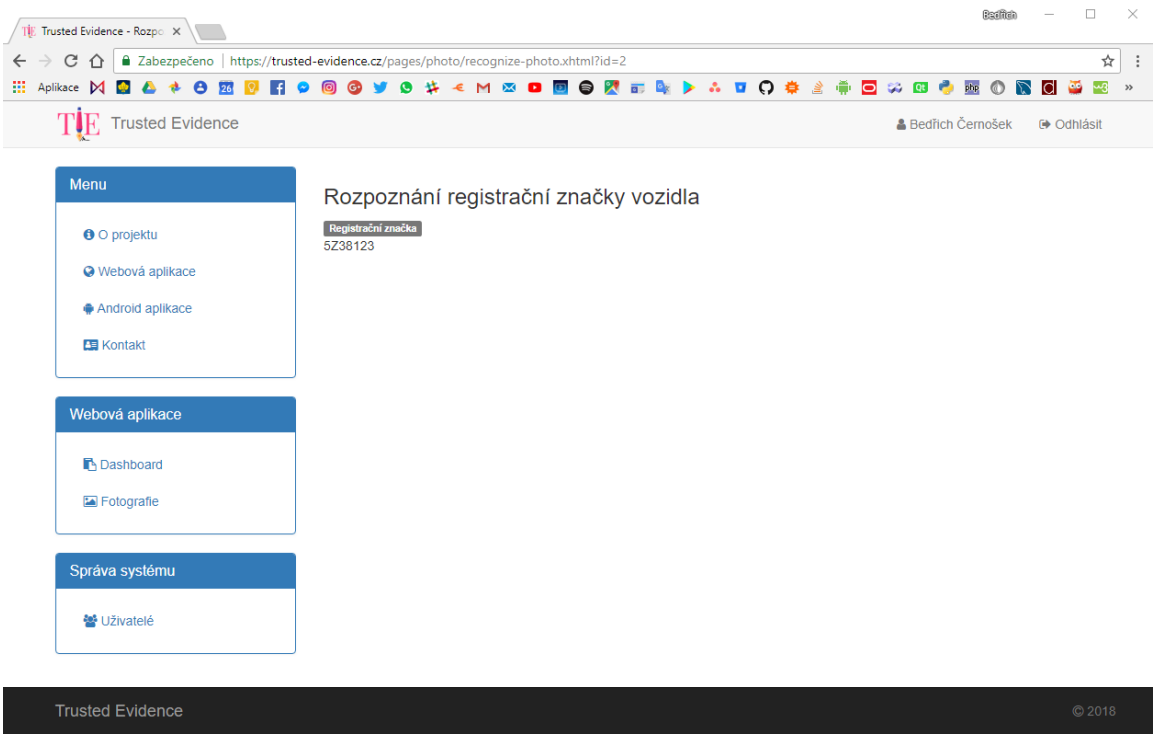
Obrázek C.4: Seznam fotografií



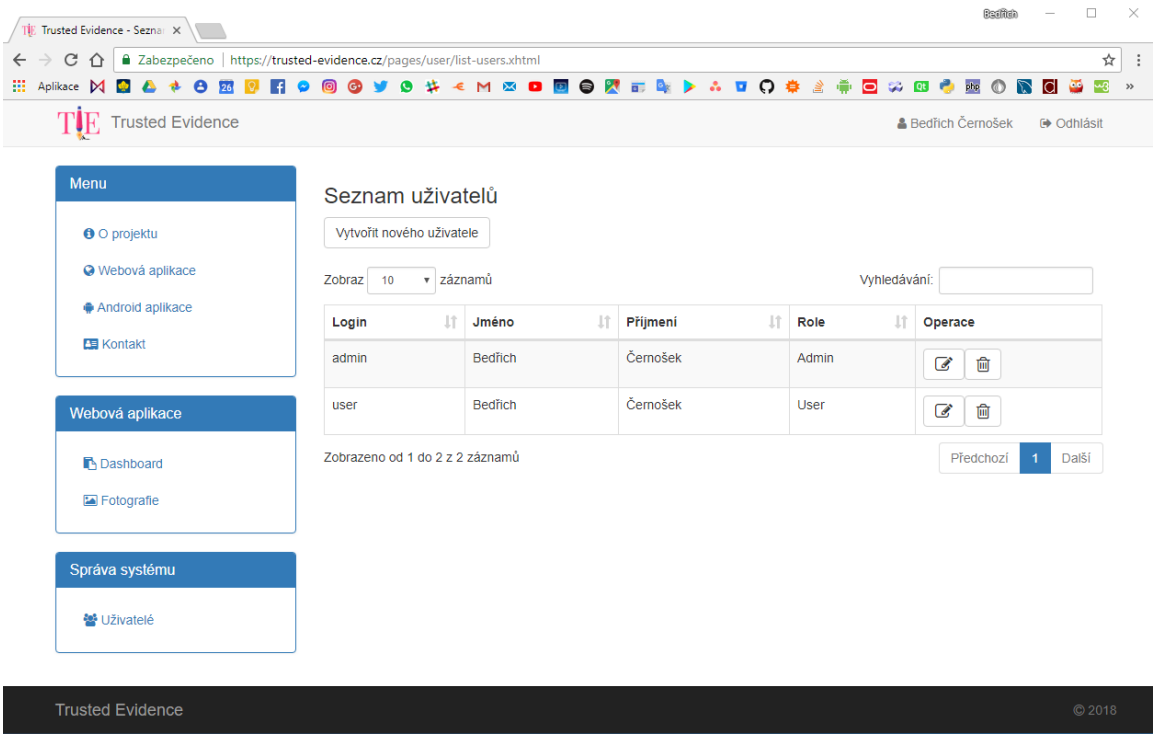
Obrázek C.5: Podrobnosti fotografie



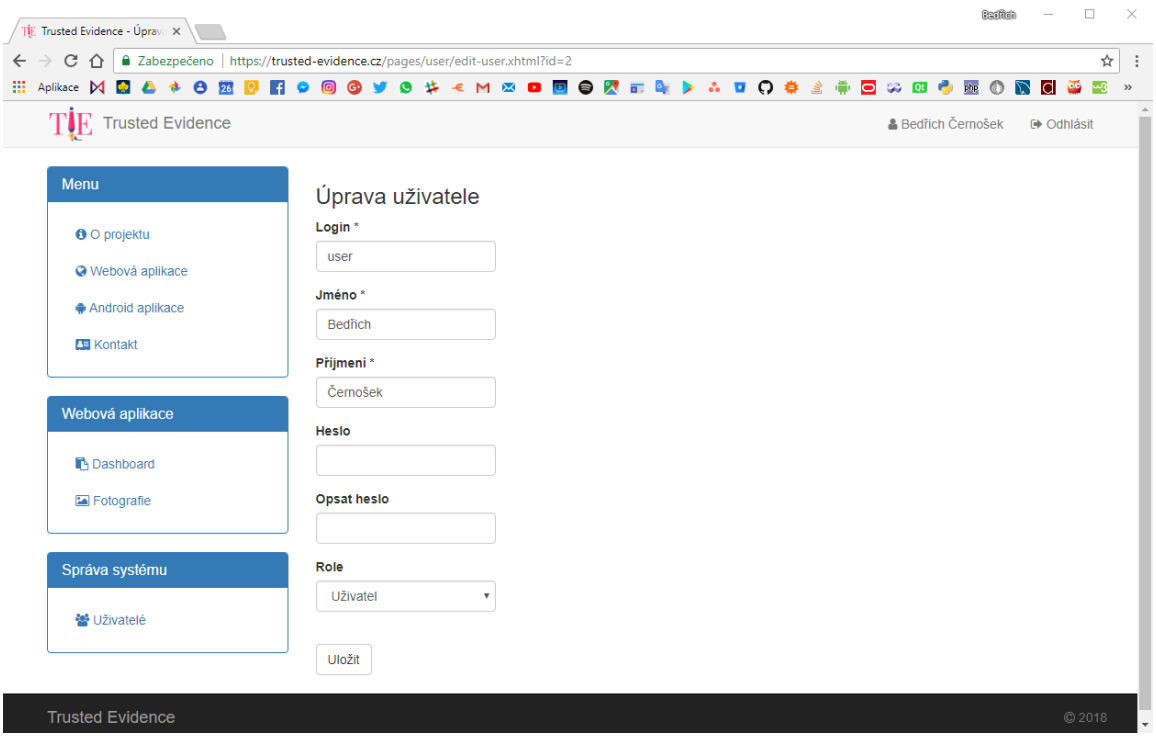
Obrázek C.6: Verifikace kryptografických prvků fotografie



Obrázek C.7: Rozpoznání registrační značky vozidla



Obrázek C.8: Seznam uživatelů



Obrázek C.9: Úprava uživatele

Příloha D

Uživatelský manuál

D.1 Klientská aplikace

Klientská mobilní aplikace Trusted Evidence se instaluje z obchodu Google Play. Aplikaci lze spustit z menu zařízení.

D.1.1 Přihlášení

Po spuštění mobilní aplikace je zobrazena hlavní obrazovka aplikace. Ve spodní části obrazovky se nachází tlačítko Přihlášení pro přístup do aplikace. Po stisku tlačítka se zobrazí obrazovka Přihlášení s přihlašovacím formulářem. V prvním řádku se zadává přihlašovací jméno uživatele tzv. login. V druhém řádku se zadává heslo. Je možné zapamatovat přihlašovací údaje pro budoucí přihlášení na používaném zařízení přepnutím přepínače u možnosti Pamatovat si mě. Stiskem tlačítka Přihlásit dochází k přihlášení. V případě úspěšného přihlášení je umožněn přístup do aplikace. V případě chyby je zobrazena chybová zpráva.

D.1.2 Registrace

Ve spodní části hlavní obrazovky se nachází tlačítko Registrace. Stiskutím tlačítka se zobrazí obrazovka Registrace s registračním formulářem. Pro registraci je nutné vyplnit všechna pole. Od vrchu se postupně vyplňuje login, e-mail, jméno, příjmení, heslo a ještě jednou heslo pro kontrolu. Registrace proběhne stiskem tlačítka Registrovat. V případě úspěšné registrace je umožněn přístup do aplikace. V případě neúspěchu je zobrazena chybová zpráva. Login a e-mail jsou unikátní vlastnosti uživatele.

D.1.3 Zapomenuté heslo

V případě zapomenutého hesla je připraveno tlačítko Zapomenuté heslo ve spodní části přihlašovacího formuláře na obrazovce Přihlášení. Postup pro zobrazení obrazovky Přihlášení je popsán v podkapitole návodu Přihlášení. Stiskem tlačítka Zapomenuté heslo je zobrazena obrazovka Zapomenuté heslo s formulářem pro zadání e-mailu. Heslo je odesláno stiskem tlačítka Odeslat heslo a je zobrazena zpráva o výsledku operace.

D.1.4 Seznam fotografií

Po úspěšném přihlášení nebo registraci je zobrazen seznam fotografií. Postup pro přihlášení je uveden v podkapitole návodu Přihlášení. Postup pro registraci je uveden v podkapitole

návodu Registrace. Seznam fotografií je dostupný z menu aplikace. Použití menu je popsáno v podkapitole návodu Menu.

D.1.5 Menu

Menu aplikace je dostupné pouze po přihlášení nebo registraci. Postup pro přihlášení je uveden v podkapitole návodu Přihlášení. Postup pro registraci je uveden v podkapitole návodu Registrace. Menu lze zobrazit pouze v hlavních obrazovkách modulů aplikace vysunutím z leva nebo stiskem tlačítka menu v horní liště.

D.1.6 Fotografování

Funkcionalita fotografování je dostupná z obrazovky seznamu fotografií. Postup pro zobrazení obrazovky je uveden v podkapitole návodu Seznam fotografií. V pravém dolním rohu obrazovky seznamu fotografií se nachází tlačítko s ikonou fotoaparátu. Stiskem tlačítka je zobrazena obrazovka fotoaparátu. Před fotografováním je vhodné provést nastavení fotoaparátu stiskem tlačítka s ikonou nastavení. Otevřena je obrazovka Nastavení fotoaparátu, kde je možné v rozevírací nabídce zvolit požadované rozlišení fotografie. Nastavení fotoaparátu se ukládá stiskem tlačítka Uložit. Po návratu zpět na obrazovku fotoaparátu lze provádět fotografování stiskem tlačítka s ikonou objektivu fotoaparátu. Výsledek operace je zobrazen v dialogu.

D.1.7 Zobrazení fotografie

Zobrazení fotografie je dostupné z obrazovky seznamu fotografií. Postup pro zobrazení obrazovky je uveden v podkapitole návodu Seznam fotografií. Zobrazit fotografii lze stiskem vybraného prvku seznamu fotografií. Alternativně lze fotografii také zobrazit otevřením kontextového menu vybraného prvku seznamu a zvolením možnosti Zobrazit.

D.1.8 Podrobnosti fotografie

Zobrazení podrobností fotografie je dostupné z obrazovky seznamu fotografií. Postup pro zobrazení obrazovky je uveden v podkapitole návodu Seznam fotografií. Podrobnosti fotografie lze zobrazit otevřením kontextového menu vybraného prvku seznamu a zvolením možnosti Podrobnosti.

D.1.9 Verifikace kryptografických prvků

Verifikace kryptografických prvků fotografie je dostupná z obrazovky seznamu fotografií. Postup pro zobrazení obrazovky je uveden v podkapitole návodu Seznam fotografií. Verifikaci kryptografických prvků lze provést otevřením kontextového menu vybraného prvku seznamu a zvolením možnosti Verifikace.

D.1.10 Detekce a rozpoznání registrační značky vozidla

Detekce a rozpoznání registrační značky vozidla na fotografii je dostupná z obrazovky seznamu fotografií. Postup pro zobrazení obrazovky je uveden v podkapitole návodu Seznam fotografií. Detekci a rozpoznání registrační značky vozidla lze provést otevřením kontextového menu vybraného prvku seznamu a zvolením možnosti Rozpoznání.

D.1.11 Smazání fotografie

Mazání fotografie je dostupné z obrazovky seznamu fotografií. Postup pro zobrazení obrazovky je uveden v podkapitole návodu Seznam fotografií. Smazat fotografii lze otevřením kontextového menu vybraného prvku seznamu a zvolením možnosti Smazat.

D.1.12 Uživatelský profil

Uživatelský profil je funkcionalita dostupná z menu aplikace pod možností Uživatel. Použití menu je popsáno v podkapitole návodu Menu. Stiskem možnosti je otevřena obrazovka Uživatel.

D.1.13 Úprava uživatele

Úprava uživatele je dostupná z obrazovky Uživatel. Postup pro zobrazení obrazovky je uveden v podkapitole Uživatelský profil. V pravém dolním rohu obrazovky Uživatel se nachází tlačítko s ikonou editace. Stiskem tlačítka je zobrazena obrazovka Úprava uživatele. Lze vytvořit novou profilovou fotografii stiskem tlačítka s ikonou fotoaparátu. Také lze vybrat existující fotografii z galerie zařízení stiskem tlačítka s ikonou fotografie. Smazat profilovou fotografii lze stiskem tlačítka s ikonou koše. Upravit lze také jméno, příjmení a e-mail. Změny jsou uloženy stiskem tlačítka Uložit. Výsledek operace je zobrazen ve zprávě.

D.1.14 Nastavení

Nastavení aplikace je funkcionalita dostupná z menu aplikace pod možností Nastavení. Použití menu je popsáno v podkapitole návodu Menu. Stiskem možnosti Nastavení je zobrazena obrazovka Nastavení s počátečním nastavením od serveru. Toto nastavení nelze měnit.

D.1.15 Klíče

Práce s RSA klíči pro digitální podpis je dostupná z menu aplikace pod možností Klíče. Použití menu je popsáno v podkapitole návodu Menu. Stiskem možnosti Klíče je zobrazena obrazovka umožňující nahrávat veřejný klíč na server stiskem tlačítka s ikonou nahrávání. Dále umožňuje exportovat klíče do externího úložiště stiskem tlačítka s ikonou exportu. Importování klíčů z externího úložiště je možné provádět stiskem tlačítka s ikonou importu.

D.1.16 Odhlášení

Odhlášení je dostupné z menu aplikace pod možností Odhlášení. Použití menu je popsáno v podkapitole návodu Menu. Stiskem možnosti Odhlášení dochází k odhlášení a zobrazení úvodní obrazovky aplikace.

D.2 Serverová aplikace

Serverová webová aplikace je dostupná na adrese https://trusted-evidence.cz pomocí webového prohlížeče.

D.2.1 Přihlášení

Přihlášení do webové aplikace lze provést kliknutím na tlačítko Přihlášení v horní liště. Alternativně se lze přihlásit také kliknutím na tlačítko Přihlášení v menu aplikace. Po kliknutí na tlačítko je zobrazena stránka s přihlašovacím formulářem. Zadává se login a heslo. Přihlášení je provedeno kliknutím na tlačítko Přihlásit. V případě úspěšného přihlášení je umožněn přístup do privátní části aplikace. V případě neúspěchu je zobrazena zpráva.

D.2.2 Registrace

Registrace ve webové aplikaci se provádí kliknutím na tlačítko Registrace v horní liště. Alternativně se lze registrovat také kliknutím na tlačítko Registrace v menu aplikace. Zadává se jméno, příjmení, login, e-mail a heslo. Registrace je provedena kliknutím na tlačítko Registrovat. Po úspěšné registraci je provedeno přesměrování na přihlašovací stránku.

D.2.3 Dashboard

Statistiky systému lze zobrazit kliknutím na tlačítko Dashboard v privátním panelu menu aplikace. Zobrazena je stránka Dashboard se statistikami systému.

D.2.4 Seznam fotografií

Seznam fotografií je dostupný kliknutím na tlačítko Fotografie v privátním panelu menu aplikace. Zobrazena je stránka Seznam fotografií.

D.2.5 Zobrazení fotografie

Zobrazení fotografie je dostupné ze stránky seznamu fotografií. Postup pro zobrazení stránky seznamu fotografií je uveden v podkapitole návodu Seznam fotografií. Zobrazení fotografie lze provést kliknutím na tlačítko prvku seznamu s ikonou fotografie ve sloupci Operace.

D.2.6 Podrobnosti fotografie

Zobrazení podrobností fotografie je dostupné ze stránky seznamu fotografií. Postup pro zobrazení stránky seznamu fotografií je uveden v podkapitole návodu Seznam fotografií. Podrobnosti fotografie lze zobrazit kliknutím na tlačítko prvku seznamu s ikonou informace ve sloupci Operace.

D.2.7 Verifikace kryptografických prvků

Verifikace kryptografických prvků fotografie je dostupná ze stránky seznamu fotografií. Postup pro zobrazení stránky seznamu fotografií je uveden v podkapitole návodu Seznam fotografií. Verifikaci kryptografických prvků lze provést kliknutím na tlačítko prvku seznamu s ikonou kontroly ve sloupci Operace.

D.2.8 Detekce a rozpoznání registrační značky vozidla

Detekce a rozpoznání registrační značky vozidla na fotografii je operace dostupná ze stránky seznamu fotografií. Postup pro zobrazení stránky seznamu fotografií je uveden v podkapitole návodu Seznam fotografií. Detekci a rozpoznání registrační značky vozidla na fotografii lze provést kliknutím na tlačítko prvku seznamu s ikonou automobilu ve sloupci Operace.

D.2.9 Smazání fotografie

Mazání fotografie je dostupné ze stránky seznamu fotografií. Postup pro zobrazení stránky seznamu fotografií je uveden v podkapitole návodu Seznam fotografií. Smazat fotografii lze kliknutím na tlačítko prvku seznamu s ikonou koše ve sloupci Operace.

D.2.10 Úprava uživatele

Úprava uživatele je dostupná z horní lišty aplikace kliknutím na tlačítko se jménem uživatele. Po kliknutí je zobrazena stránka Úprava uživatele. Je možné upravit jméno, příjmení a heslo. Změny jsou uloženy kliknutím na tlačítko Uložit. Výsledek operace je zobrazen ve zprávě.

D.2.11 Odhlášení

Odhlášení z webové aplikace je možné provést kliknutím na tlačítko Odhlásit v horní liště.

Příloha E

Dokumentace API

E.1 Uživatel

Požadavky související s objektem uživatele jsou zpracovány pomocí služby UserRestService. Při využití služby je aplikován autentizační filtr AuthRestFilter a filtr rolí AllowedRolesRestFilter.

E.1.1 Přihlášení

Metoda:	`GET`	
Cesta:	`/user/login`	
Hlavička:	`Authorization`	Basic Auth
Parametry:	`String login`	
Návrat:	`InitialSetup setup`	JSON

Požadavek slouží k autentizaci existujícího uživatele. Pomocí parametru uživatelského jména je vyhledán v databázi požadovaný objekt uživatele. Je provedena kontrola identifikátoru objektu uživatele porovnáním s identifikátorem objektu uživatele z autentizace. Výstupem je počáteční nastavení od serveru obsahující objekt uživatele.

E.1.2 Registrace

Metoda:	`POST`	
Cesta:	`/user/signup`	
Tělo:	`User user`	JSON
Návrat:	`InitialSetup setup`	JSON

Požadavek slouží k registraci nového uživatele. Dochází k vytvoření nového objektu uživatele v databázi. Pokud se již objekt uživatele s daným loginem nebo e-mailem vyskytuje v databázi, pak je vrácena chybová zpráva v návratovém objektu. Výstupem je počáteční nastavení od serveru.

E.1.3 Zapomenuté heslo

Metoda:	`POST`	
Cesta:	`/user/lost`	
Tělo:	`String mail`	Plain text
Návrat:	`Response response`	

Požadavek slouží pro zaslání zapomenutých přihlašovacích údajů. Objekt uživatele je vyhledán v databázi pomocí e-mailu z požadavku. Server odesílá uživateli e-mail s přihlašovacími údaji. Výstupem je objekt třídy Response.

E.1.4 Získání objektu uživatele

Metoda:	`GET`	
Cesta:	`/user/get/me`	
Hlavička:	`Authorization`	Basic Auth
Parametry:	`long id_user`	
Návrat:	`User user`	JSON

Požadavek slouží pro získání objektu uživatele. Objekt uživatele je získán z databáze pomocí identifikátoru objektu uživatele z parametru požadavku. Identifikátor objektu uživatele je kontrolován porovnáním s identifikátorem objektu uživatele z autentizace.

E.1.5 Získání objektu uživatele administrátorem

Metoda:	`GET`	
Cesta:	`/user/get`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Parametry:	`long id_user`	
Návrat:	`User user`	JSON

Požadavek slouží pro získání objektu uživatele administrátorem. Objekt uživatele je získán z databáze pomocí identifikátoru objektu uživatele z parametru požadavku. Kontrolována je role uživatele pomocí filtru rolí.

E.1.6 Seznam objektů uživatelů

Metoda:	`GET`	
Cesta:	`/user/list`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Návrat:	`List<User> users`	JSON

Požadavek slouží pro získání seznamu objektů uživatelů administrátorem. Z databáze jsou získány všechny objekty uživatelů. Kontrolována je role uživatele pomocí filtru rolí.

E.1.7 Uložení objektu uživatele

Metoda:	`POST`	
Cesta:	`/user/post/me`	
Hlavička:	`Authorization`	Basic Auth
Tělo:	`User user`	JSON
Návrat:	`String result`	JSON

Požadavek slouží pro uložení objektu uživatele. Přijatý objekt uživatele je aktualizován v databázi. Je kontrolován identifikátor objektu uživatele porovnáním s identifikátorem objektu uživatele z autentizace. Výstupem je řetězec informující o výsledku operace.

E.1.8 Uložení objektu uživatele administrátorem

Metoda:	`POST`	
Cesta:	`/user/post`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Tělo:	`User user`	JSON
Návrat:	`String result`	JSON

Požadavek slouží pro uložení objektu uživatele administrátorem. Přijatý objekt uživatele je uložen do databáze. Pokud objekt neexistuje, tak je vytvořen. Pokud objekt v databázi již existuje, tak je aktualizován. Je kontrolována role uživatele pomocí filtru rolí. Výstupem je řetězec informující o výsledku operace.

E.1.9 Uložení veřejného klíče

Metoda:	`POST`	
Cesta:	`/user/key/my`	
Hlavička:	`Authorization`	Basic Auth
Tělo:	`Keys keys`	JSON
Návrat:	`Response response`	

Požadavek slouží pro uložení veřejného RSA klíče na serveru k ověřování digitálního podpisu ve webové aplikaci. Veřejný klíč je uložen v databázi do objektu autentizovaného uživatele. Výstupem je objekt třídy Response.

E.2 Fotografie

Požadavky související s objektem fotografie a souborem fotografie jsou zpracovány pomocí služby PhotoRestService. Při využití služby je aplikován autentizační filtr AuthRestFilter a filtr rolí AllowedRolesRestFilter.

E.2.1 Nahrání fotografie na server

Metoda:	`POST`	
Cesta:	`/photo/upload/object`	
Hlavička:	`Authorization`	Basic Auth
Tělo:	`InputStream file`	Multipart form data
	`FormDataBodyPart part`	Multipart form data
Návrat:	`String file_location`	JSON

Požadavek slouží pro nahrání souboru fotografie a objektu fotografie na server. Je kontrolován identifikátor objektu uživatele z objektu fotografie porovnáním s identifikátorem objektu uživatele z autentizace. Soubor fotografie je po přijetí uložen do úložiště na serveru. Do objektu fotografie je vložena cesta k uloženému souboru fotografie. Dále je získáno časové razítko od důvěryhodného serveru. Časové razítko a token je vložen do objektu fotografie, který je následně uložen v databázi. Výstupem je řetězec s cestou k uloženému souboru fotografie na serveru.

E.2.2 Získání objektu fotografie

Metoda:	`GET`	
Cesta:	`/photo/get/my`	
Hlavička:	`Authorization`	Basic Auth
Parametr:	`long id_photo`	
Návrat:	`Photo photo`	JSON

Požadavek slouží pro získání objektu fotografie. Objekt fotografie je získán z databáze pomocí identifikátoru objektu fotografie z parametru požadavku. Je kontrolován identifikátor objektu uživatele z objektu fotografie porovnáním s identifikátorem objektu uživatele z autentizace.

E.2.3 Získání objektu fotografie administrátorem

Metoda:	`GET`	
Cesta:	`/photo/get`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Parametr:	`long id_photo`	
Návrat:	`Photo photo`	JSON

Požadavek slouží pro získání objektu fotografie administrátorem. Objekt fotografie je získán z databáze pomocí identifikátoru objektu fotografie z parametru požadavku. Je prováděna kontrola role uživatele pomocí filtru rolí.

E.2.4 Získání seznamu objektů fotografií

Metoda:	`GET`	
Cesta:	`/photo/list/my`	
Hlavička:	`Authorization`	Basic Auth
Návrat:	`List<Photo> photo`	JSON

Požadavek slouží pro získání seznamu objektů fotografií. Je získán z databáze seznam objektů fotografií s identifikátorem objektu uživatele shodým s identifikátorem objektu uživatele z autentizace. Výstupem je seznam objektů fotografií uživatele.

E.2.5 Získání seznamu objektů fotografií administrátorem

Metoda:	`GET`	
Cesta:	`/photo/list`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Návrat:	`List<Photo> photo`	JSON

Požadavek slouží pro získání seznamu objektů fotografií administrátorem. Výstupem je seznam všech objektů fotografií z databáze. Je prováděna kontrola role uživatele pomocí filtru rolí.

E.2.6 Uložení objektu fotografie

Metoda:	`POST`	
Cesta:	`/photo/post/my`	
Hlavička:	`Authorization`	Basic Auth
Parametr:	`long id_photo`	
Návrat:	`Response response`	

Požadavek slouží pro uložení objektu fotografie. Přijatý objekt fotografie je uložen v databázi. Je kontrolován identifikátor objektu uživatele z objektu fotografie porovnáním s identifikátorem objektu uživatele z autentizace. Výstupem je objekt třídy Response.

E.2.7 Uložení objektu fotografie administrátorem

Metoda:	`POST`	
Cesta:	`/photo/post`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Parametr:	`long id_photo`	
Návrat:	`Response response`	

Požadavek slouží pro uložení objektu fotografie administrátorem. Přijatý objekt fotografie je uložen v databázi. Je kontrolována role uživatele pomocí filtru rolí. Výstupem je objekt třídy Response.

E.2.8 Smazání fotografie

Metoda:	`DELETE`	
Cesta:	`/photo/delete/my`	
Hlavička:	`Authorization`	Basic Auth
Parametr:	`long id_photo`	
Návrat:	`Response response`	

Požadavek slouží pro smazání souboru fotografie a objektu fotografie. Je získán objekt z databáze pomocí identifikátoru objektu fotografie z parametru požadavku. Je kontrolován identifikátor objektu uživatele z objektu fotografie porovnáním s identifikátorem objektu uživatele z autentizace. Je smazán soubor fotografie v úložišti a je smazán objekt fotografie v databázi. Výstupem je objekt třídy Response.

E.2.9 Smazání fotografie administrátorem

Metoda:	`DELETE`	
Cesta:	`/photo/delete`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Parametr:	`long id_photo`	
Návrat:	`Response response`	

Požadavek slouží pro smazání souboru fotografie a objektu fotografie administrátorem. Je získán objekt z databáze pomocí identifikátoru objektu fotografie z parametru požadavku. Je kontrolována role uživatele pomocí filtru rolí. Je smazán soubor fotografie v úložišti a je smazán objekt fotografie v databázi. Výstupem je objekt třídy Response.

E.2.10 Verifikace kryptografických prvků fotografie

Metoda:	`POST`	
Cesta:	`/photo/verify/my`	
Hlavička:	`Authorization`	Basic Auth
Parametr:	`long id_photo`	
Tělo:	`Keys keys`	JSON
Návrat:	`Verification verification`	JSON

Požadavek slouží pro ověření kryptografických prvků fotografie. Je získán objekt fotografie pomocí identifikátoru objektu fotografie z parametru požadavku. Je kontrolován identifikátor uživatele z objektu fotografie porovnáním s identifikátorem objektu uživatele z autentizace. Je ověřena platnost kryptografických prvků fotografie a výsledek vložen do vytvořeného objektu. Výstupem je objekt třídy Verification.

E.2.11 Verifikace kryptografických prvků fotografie administrátorem

Metoda:	`POST`	
Cesta:	`/photo/verify`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Parametr:	`long id_photo`	
Tělo:	`Keys keys`	JSON
Návrat:	`Verification verification`	JSON

Požadavek slouží pro ověření kryptografických prvků fotografie administrátorem. Je získán objekt fotografie pomocí identifikátoru objektu fotografie z parametru požadavku. Je kontrolována role uživatele pomocí filtru rolí. Je ověřena platnost kryptografických prvků fotografie a výsledek vložen do vytvořeného objektu. Výstupem je objekt třídy Verification.

E.2.12 Detekce a rozpoznání registrační značky vozidla

Metoda:	`GET`	
Cesta:	`/photo/recognize/my`	
Hlavička:	`Authorization`	Basic Auth
Parametr:	`long id_photo`	
Návrat:	`String plate`	JSON

Požadavek slouží pro detekci a rozpoznání registrační značky vozidla na fotografii. Je získán objekt fotografie pomocí identifikátoru objektu fotografie z parametru požadavku. Je kontrolován identifikátor objektu uživatele z objektu fotografie porovnáním s identifikátorem objektu uživatele z autentizace. Je detekována a rozpoznána registrační značka vozidla na fotografii z úložiště. Výstupem je řetězec s registrační značkou vozidla.

E.2.13 Detekce a rozpoznání registrační značky vozidla administrátorem

Metoda:	`GET`	
Cesta:	`/photo/recognize`	
Hlavička:	`Authorization`	Basic Auth
Role:	`UserRole Admin`	
Parametr:	`long id_photo`	
Návrat:	`String plate`	JSON

Požadavek slouží pro detekci a rozpoznání registrační značky vozidla na fotografii administrátorem. Je získán objekt fotografie pomocí identifikátoru objektu fotografie z parametru požadavku. Je kontrolována role uživatele pomocí filtru rolí. Je detekována a rozpoznána registrační značka vozidla na fotografii z úložiště. Výstupem je řetězec s registrační značkou vozidla.